


```

LL               IIIIII             SSSSSSSS
LL               IIIIII             SSSSSSSS
LL               II                 SS
LL               II                 SS
LL               II                 SS
LL               II                 SS
LL               II                 SSSSSS
LL               II                 SSSSSS
LL               II                 SS
LL               II                 SS
LL               II                 SS
LL               II                 SS
LLLLLLLLLLLLLL  IIIIII             SSSSSSSS
LLLLLLLLLLLLLL  IIIIII             SSSSSSSS

```

(1)	454	TM78/TU78 MASSBUS REGISTER OFFSETS
(1)	573	TM78/TU78 DEVICE DEPENDENT UNIT CONTROL BLOCK OFFSETS
(1)	752	Hardware function and interrupt codes
(1)	1027	TM78/TU78 FUNCTION DECISION TABLE
(1)	1144	CANCEL I/O ON CHANNEL
(1)	1217	START I/O OPERATION (and RESTARTIO)
(1)	1415	Start NOP, DRVCLR, PACKACK, SENSECHAR or SENSEMODE I/O functions
(1)	1471	Start REWIND, RECAL, READPRESET, REWINDOFF or UNLOAD function.
(1)	1556	Start a SETCHAR or a SETMODE function
(1)	1671	Start a SKIPFILE or a SPACEFILE function
(1)	1822	Start a SKIPRECORD or a SPACERECORD function
(1)	2038	COMPUTE NDT REPEAT subroutine
(1)	2067	Start WRITEOF or WRITEMARK or DSE function
(1)	2184	NDT Function Executor.
(1)	2287	Start READBLK or READPBLK functions.
(1)	2344	Start WRITELBLK or WRITEPBLK functions.
(1)	2381	Start WRITECHECK function.
(1)	2450	READ TRANSFER ACTION TABLE
(1)	2513	READ OR WRITECHECK BLOCK
(1)	2763	WRITE TRANSFER ACTION TABLE
(1)	2815	WRITE BLOCK
(1)	2983	LOAD MBA REGISTERS
(1)	3075	INTERNAL SPACE
(1)	3130	TIMEOUT POWERFAIL
(1)	3392	REWIND ROUTINE
(1)	3485	Device Error Routine.
(1)	3512	Invoke Extended Sense routine
(1)	3628	AWAIT MANUAL REWIND
(1)	3677	Function Completion Common Exit
(1)	3731	TM78/TU78 REGISTER DUMP ROUTINE
(1)	3758	TM78 CONTROLLER INITIALIZATION
(1)	3863	TM78-TU78 TAPE DRIVE INITIALIZATION
(1)	3999	TM78/TU78 UNSOLICITED INTERRUPT PROCESSING
(1)	4111	TM78 Classify Drive Type and set parameters.
(1)	4148	RECORD SENSE INFO
(1)	4307	TM78 NOTREADY (PRIOR and POST)
(1)	4354	FAULT INTERRUPT
(1)	4441	AWAIT-TMRDY
(1)	4480	ISSUE-TMCLR
(1)	4564	TM78/TU78 SLAVE CONTROLLER INTERRUPT DISPATCHER

```
0000 1 .TITLE TFDRIVER - TM78/TU78 MAGTAPE DRIVER
0000 2 .IDENT 'V04-000'
0000 3
0000 4
0000 5 *****
0000 6 *
0000 7 * COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
0000 8 * DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
0000 9 * ALL RIGHTS RESERVED.
0000 10 *
0000 11 * THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
0000 12 * ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
0000 13 * INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
0000 14 * COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
0000 15 * OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
0000 16 * TRANSFERRED.
0000 17 *
0000 18 * THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
0000 19 * AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
0000 20 * CORPORATION.
0000 21 *
0000 22 * DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
0000 23 * SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
0000 24 *
0000 25 *
0000 26 *****
0000 27
0000 28 Robert Rappaport 29-February-1980
0000 29
0000 30 MODIFIED B:
0000 31
0000 32 V03-018 MMD0310 Meg Dumont, 19-Jul-1984 9:33
0000 33 Add support for the UCBSL_MEDIA_ID field
0000 34
0000 35 V03-017 RAS0300 Ron Schaefer 27-Apr-1984
0000 36 Add DEV$M_NNM characteristic to DECHAR2 so that these
0000 37 devices will have the 'node$' prefix.
0000 38
0000 39 V03-016 MMD0221 Meg Dumont, 9-Jan-1984 17:05
0000 40 Fixed SKIPRECORD to set MT$M_EOF when TM is encountered.
0000 41
0000 42 V03-015 ROW0258 Ralph O. Weber 17-NOV-1983
0000 43 The Paul Painter Memorial Enhancement
0000 44 Named for one of the unfortunate customers who suffered much
0000 45 to determine the great UCBSL_MT_RECORD secret while trying to
0000 46 create a user-written magtape driver, this change eliminates
0000 47 use of the device dependent field, UCBSL_MF_RECORD in favor of
0000 48 the device independent field, UCBSL_RECORD.
0000 49
0000 50 V03-014 MMD0203 Meg Dumont, 9-Nov-1983 19:18
0000 51 Fixed a bug in SUBCODE_RUT handling where the routines address
0000 52 didn't get relocated properly.
0000 53
0000 54 V03-013 MMD0201 Meg Dumont, 7-Sep-1983 15:09
0000 55 Increase TU78 maximum timeouts to include 3600' tapes. This
0000 56 includes changing the constants TU78_MAX_REWIND, TU78_MAX_DSE
0000 57 and TU78_MAX_SPACE
```

```
0000 58 :  
0000 59 :  
0000 60 : V03-012 ROW0213 Ralph O. Weber 23-AUG-1983  
0000 61 : Change basing for device-dependent UCB from UCB$$_DP_LINK+4 to  
0000 62 : a field independent UCB$$_LCL_TAPE_LENGTH. This allows the  
0000 63 : device-independent UCB to be altered without having to edit  
0000 64 : this module.  
0000 65 :  
0000 66 : V03-011 MMD0198 Meg Dumont, 19-Aug-1983 13:16  
0000 67 : Added code in CONTROL_INIT and UNIT_INIT to test the  
0000 68 : rev level of the microcode. This is needed because  
0000 69 : the TM78 controller has a know problem where it might  
0000 70 : lose data at EOT, therefore we want to let customers  
0000 71 : know they may run into problems. When the ucode is  
0000 72 : available to the field then we will not configure in  
0000 73 : then devices until they are brought up to rev.  
0000 74 :  
0000 75 : V03-010 RLRDPATH1 Robert L. Rappaport 31-May-1983  
0000 76 : Allow UCB to include new DUAL PORT extension by  
0000 77 : changing base of where we begin the private TFDRIVER  
0000 78 : extension from UCB$$_DPC+4 to UCB$$_DP_LINK+4.  
0000 79 :  
0000 80 : V03-009 RLR50603 Robert L. Rappaport 5-Jan-1983  
0000 81 : When reading, do a SENSE command when we are positioned  
0000 82 : after the first record after the read completes.  
0000 83 :  
0000 84 : V03-008 RLRFAULT Robert L. Rappaport 11-Nov-1982  
0000 85 : Allow TM Clears every 15 seconds (instead of 10 minutes).  
0000 86 :  
0000 87 : V03-007 RLR48869a Robert L. Rappaport 11-Nov-1982  
0000 88 : Add additional checking to be able to tolerate return of  
0000 89 : zero length in MF_BC register in read opposite.  
0000 90 :  
0000 91 : V03-006 RLR48869 Robert L. Rappaport 29-Oct-1982  
0000 92 : Correct error in READ OPPOSITE that used wrong register.  
0000 93 :  
0000 94 : V03-005 RLR50396 Robert L. Rappaport 22-Oct-1982  
0000 95 : If at BOT, have RECORD SENSE_INFO not update density in  
0000 96 : UCB$$_DEVDEPEND. In this way the last user specified  
0000 97 : density prevails.  
0000 98 :  
0000 99 : V03-004 KDM0002 Kathleen D. Morse 28-Jun-1982  
0000 100 : Added $DYNDEF, $PRDEF, and $SSDEF.  
0000 101 :  
0000 102 : V03-003 RLRECO01 Robert L. Rappaport 17-June-1982  
0000 103 : 1. Make AVAILABLE Buffered I/O in FDT.  
0000 104 : 2. Change BLSS to BGEQU in ISSUE_TMCLR.  
0000 105 :  
0000 106 : TM78/TU78 MAGTAPE DRIVER  
0000 107 :  
0000 108 :  
0000 109 : MACRO LIBRARY CALLS  
0000 110 :  
0000 111 :  
0000 112 : $ADPDEF ;Define ADP offsets.  
0000 113 : $CRBDEF ;Define CRB offsets  
0000 114 : $DDBDEF ;Define DDB offsets
```

```
0000 115      $DEVDEF      ;Define DEVICE CHARACTERISTICS bits
0000 116      $DPTDEF      ;Define DPT offsets
0000 117      $DYNDEF      ;Define dynamic data structure types
0000 118      $EMBDEF      ;Define EMB offsets
0000 119      $IDBDEF      ;Define IDB offsets
0000 120      $IODEF       ;Define I/O FUNCTION codes
0000 121      $IRPDEF      ;Define IRP offsets
0000 122      $MBADEF      ;Define MBA REGISTER offsets
0000 123      $MTDEF       ;Define MAGTAPE STATUS bits
0000 124      $PRDEF       ;Define processor registers
0000 125      $SSDEF       ;Define system status codes
0000 126      $UCBDEF      ;Define UCB offsets
0000 127      $VECDEF      ;Define INTERRUPT DISPATCH VECTOR offsets
0000 128      $WCBDEF      ;Define WCB offsets
0000 129
0000 130 :
0000 131 :
0000 132 : Local Macros
0000 133 :
0000 134 :
0000 135 :
0000 136 : SUBSAVE - macro to save the return point of a subroutine in the SUBSTACK
0000 137 : located in the UCB. The SUBSTACK is a mechanism that allows for
0000 138 : saving several nested return points in a context where data cannot
0000 139 : be left on the normal stack; such as when IOFORKS or WFIKPC macros
0000 140 : are executed. The SUBSTACK is implemented by allocating an array of
0000 141 : longwords in the UCB known as UCB$$_MF_SUBSTACK and by also providing
0000 142 : another longword in the UCB known as UCB$$_MF_SUBSP which is used to
0000 143 : point to the next available longword in the SUBSTACK array. At
0000 144 : STARTIO time we initialize UCB$$_MF_SUBSP to point to the base of this
0000 145 : array and from then on the SUBSAVE, SUBRETURN, SUBPUSH and SUBPOP
0000 146 : macro invocations keep this pointer upto date.
0000 147 :
0000 148 :
0000 149 : .MACRO SUBSAVE
0000 150 : POPL @UCB$$_MF_SUBSP(R5) ; Save return on SUBSTACK.
0000 151 : ADDL #4,UCB$$_MF_SUBSP(R5) ; Update SUBSTACK pointer.
0000 152 : .ENDM SUBSAVE
0000 153 :
0000 154 :
0000 155 : SUBRETURN - macro to pop the next value off the SUBSTACK and return control
0000 156 : to this address.
0000 157 :
0000 158 :
0000 159 : .MACRO SUBRETURN
0000 160 : SUBL #4,UCB$$_MF_SUBSP(R5) ; Set SUBSTACK ptr to last used entry.
0000 161 : PUSHL @UCB$$_MF_SUBSP(R5) ; Pop return from SUBSTACK to stack.
0000 162 : RSB ; Return to caller.
0000 163 : .ENDM SUBRETURN
0000 164 :
0000 165 :
0000 166 :
0000 167 : SUBPUSH - macro to push upto ten LONGWORDS onto the SUBSTACK.
0000 168 :
0000 169 :
0000 170 : .MACRO SUBPUSH LW0,LW1,LW2,LW3,LW4,LW5,LW6,LW7,LW8,LW9
0000 171 : .NARG ..X..
```

```
0000 172 ..Y..=0
0000 173 .IRP LONGWORD,<LW0,LW1,LW2,LW3,LW4,LW5,LW6,LW7,LW8,LW9>
0000 174 .IF LT ..Y..-..X.. ; Do only for explicit args.
0000 175 ..Y..=..Y..+1
0000 176 .IF NOT BLANK LONGWORD
0000 177 MOVE LONGWORD,@UCB$L_MF_SUBSP(R5) ; Move value to SUBSTACK
0000 178 .IFF
0000 179 CLRL @UCB$L_MF_SUBSP(R5) ; Clear SUBSTACK entry.
0000 180 .ENDC
0000 181
0000 182 ADDL #4,UCB$L_MF_SUBSP(R5) ; Update SUBSTACK pointer.
0000 183 .ENDC
0000 184 .ENDR
0000 185 .ENDM SUBPUSH
0000 186
0000 187 :
0000 188 : SUBPOP - macro to pop upto ten LONGWORD values off the SUBSTACK.
0000 189 :
0000 190
0000 191 .MACRO SUBPOP LW0,LW1,LW2,LW3,LW4,LW5,LW6,LW7,LW8,LW9
0000 192 .NARG ..X..
0000 193 ..Y..=0
0000 194 .IRP LONGWORD,<LW0,LW1,LW2,LW3,LW4,LW5,LW6,LW7,LW8,LW9>
0000 195 .IF LT ..Y..-..X.. ; Do only for explicit args.
0000 196 ..Y..=..Y..+1
0000 197 SUBL #4,UCB$L_MF_SUBSP(R5) ; Update SUBSTACK pointer.
0000 198 .IF NOT BLANK LONGWORD
0000 199 MOVE @UCB$L_MF_SUBSP(R5),LONGWORD ; Move value from SUBSTACK
0000 200 .ENDC
0000 201 .ENDR
0000 202 .ENDM SUBPOP
0000 203
0000 204 :
0000 205 : REWIND - macro to invoke REWIND ROUTINE. Invoked in normal START_REWIND
0000 206 : logic and also in POWERFAIL TIMEOUT routine. The macro prepares a
0000 207 : call to REWIND ROUTINE by placing the needed arguments in R0 and R1.
0000 208 : The calling sequence provides for a REWIND command (be it UNLOAD or
0000 209 : REWIND) to be passed in R0, and a bitmask to be passed in R1. The
0000 210 : only bit of interest in R1 is IOSM_NOWAIT, which instructs REWIND_ROUTINE
0000 211 : as to whether it should wait for the rewind to terminate.
0000 212 : It also provides for an error return.
0000 213 :
0000 214 :
0000 215 .MACRO REWIND CMD,WAIT,ERROR_LABEL
0000 216 .IF B CMD
0000 217 MOVZBL #F_REWIND!GO_BIT,R0 ; If no CMD specified, assume REWIND
0000 218 .IFF
0000 219 .IF DIFFERENT R0,<CMD>
0000 220 MOVZBL CMD,R0 ; If specified, and different from
0000 221 .ENDC ; R0, then move it to R0.
0000 222 .ENDC
0000 223
0000 224 .IF IDENTICAL YES,<WAIT>
0000 225 CLRL R1 ; Clear IOSM_NOWAIT bit in R1.
0000 226 .IFF
0000 227 .IF IDENTICAL NO,<WAIT>
0000 228 MOVZWL #IOSM_NOWAIT,R1 ; Set IOSM_NOWAIT bit in R1.
```

```
0000 229 .IFF
0000 230 .IF DIFFERENT R1,<WAIT>
0000 231 MOVZWL WAIT,R1 ; Move specified mask to R1.
0000 232 .ENDC
0000 233 .ENDC
0000 234 .ENDC
0000 235
0000 236 BSBW REWIND ROUTINE ; Call REWIND ROUTINE.
0000 237 .WORD ERROR_LABEL-2 ; Relative address of ERROR_LABEL.
0000 238 .ENDM REWIND
0000 239
0000 240 :
0000 241 : Two macros to help coordinate with asynchronous 'B' type fault attention
0000 242 : interrupts and the actions we take as a result of them. TEST_TMRDY
0000 243 : is invoked prior to loading final device registers and setting the
0000 244 : GO bit. POST TEST_TMRDY is invoked after awakening from a WFI/KPCH.
0000 245 : Briefly they test to make sure nothing horrible has happened to the
0000 246 : TM78 controller. If either finds a problem, it branches to
0000 247 : a special piece of code which tries to recover as best it can.
0000 248 :
0000 249 :
0000 250 .MACRO PRIOR_TEST_TMRDY ?LO
0000 251 BITW #UCBSW_MF_TMRDY!- ; Is TMRDY or an ATTN
0000 252 UCBSW_MF_ATTN- ; pending?
0000 253 UCBSW_DEVSTS(R5)
0000 254 BEQL LO ; If NOT, branch around.
0000 255 BRW TM78_NOTREADY_PRIOR
0000 256 LO:
0000 257 .ENDM PRIOR_TEST_TMRDY
0000 258
0000 259 .MACRO POST_TEST_TMRDY ?LO
0000 260 BITW #UCBSW_MF_TMRDY!- ; Is TMRDY or an ATTN
0000 261 UCBSW_MF_ATTN- ; pending?
0000 262 UCBSW_DEVSTS(R5)
0000 263 BEQL LO ; If NOT, branch around.
0000 264 BRW TM78_NOTREADY_POST
0000 265 LO:
0000 266 .ENDM POST_TEST_TMRDY
0000 267
0000 268 :
0000 269 : EX_NDT_CMD - Macro to EXECUTE a NDT command.
0000 270 : Macro accepts four parameters:
0000 271 : 1. CMD - the symbolic name of the NDT command.
0000 272 : 2. REPEAT - the repeat count to be loaded in the register.
0000 273 : 3. TIMEOUT - the number of seconds to allow for TIMEOUT for
0000 274 : this command.
0000 275 : 4. ERROR_LABEL - the label to which control should be
0000 276 : transferred in the event a call to TIMEOUT_POWERFAIL
0000 277 : should experience return.
0000 278 :
0000 279 : The macro loads UCBSW_MF_NDTCR with the template command including
0000 280 : the repeat count. It also loads UCBSW_MF_TIMEOUT with the
0000 281 : value passed as the TIMEOUT parameter. It then calls the common
0000 282 : NDT command executor subroutine NDT_EXECUTOR.
0000 283 :
0000 284 :
0000 285 .MACRO EX_NDT_CMD CMD,REPEAT,TIMEOUT,ERROR_LABEL
```



```
0000 286 .NTYPE .XX.,CMD ; Determine if operand is register.
0000 287 .XX.=.XX.@-4&^XF
0000 288 .IF EQUAL .XX.-5 ; If it is a register then...
0000 289
0000 290 MOVZBW CMD,- ; Move command code and GO BIT from register
0000 291 UCB$W_MF_NDTCR(R5) ; to NDT command template in UCB.
0000 292 .IFF ; If operand is NOT a register then...
0000 293 MOVZBW MF 'CMD'!GO BIT,- ; Move literal command code + GO BIT
0000 294 UCB$W_MF_NDTCR(R5) ; to NDT command template in UCB.
0000 295 .ENDC
0000 296 .IF NB REPEAT ; Only if we have an explicit REPEAT.
0000 297 ASSUME MF_NDTCR_V_CCNT EQ 8
0000 298 ASSUME MF_NDTCR_S_CCNT EQ 8
0000 299 MOVB REPEAT,-
0000 300 UCB$W_MF_NDTCR+1(R5) ; Move explicit REPEAT count to template.
0000 301 .ENDC
0000 302
0000 303 .IF NB TIMEOUT ; Only assemble following if TIMEOUT
0000 304 ; is specified.
0000 305 MOVL TIMEOUT,-
0000 306 UCB$W_MF_TIMEOUT(R5) ; Remember TIMEOUT spec in UCB.
0000 307 .ENDC
0000 308
0000 309 BSBW NDT_EXECUTOR ; Call common NDT Executor subroutine.
0000 310 .WORD ERROR_LABEL-.-2 ; Relative address of ERROR_LABEL.
0000 311 .ENDM EX_ND_CMD
0000 312
0000 313
0000 314 $DEFINI TAT ; Transfer Action Table entry layout.
0000 315
0000 316 $DEF TAT_HIC .BLKB 1 ; Hardware Interrupt Code.
0001 317 $DEF TAT_SOFT_STAT .BLKW 1 ; Software status returned to user.
0003 318 $DEF TAT_DEVDEPEND .BLKW 1 ; Bits to set in UCB$W_DEVDEPEND+2.
0005 319 $DEF TAT_FLAGS .BLKB 1 ; Flags that determine actions.
0006 320 $DEF TAT_SUBCODE_RUT .BLKL 1 ; Routine to deal with error subcodes.
000A 321
000A 322 _VIELD TAT,0,<-
000A 323 <ERRLOG,,M>,- ; If set then we perform error logging.
000A 324 <POSITION,,M>,- ; If set then we update UCB$W_RECORD.
000A 325 <COUNT,,M>,- ; If set then we return UCB$W_MF_BC in R0.
000A 326 <PREVTM,,M>,- ; If set then we update UCB$W_MF_PREVTM.
000A 327 >
000A 328
000A 329 TAT_LENGTH=.
000A 330
000A 331 $DEFEND TAT
0000 332 :
0000 333 : TAT_ENTRY - macro to declare a Transfer Action Table entry. These
0000 334 : entries serve as a decision table which determines actions to
0000 335 : take after executing a data transfer command. The first
0000 336 : byte of the TAT entry contains a possible Hardware Interrupt
0000 337 : Code (HIC). The next word contains the Software status that
0000 338 : we return to the user corresponding to this HIC. The next
0000 339 : word contains a bit mask which we OR into the high word of
0000 340 : UCB$W_DEVDEPEND when we get this HIC. Finally the last byte
0000 341 : is a bit mask that determines certain actions to take. Bits
0000 342 : defined in this mask determine whether we ERROR LOG the condition;
```

```
0000 343 : whether we return the byte count of bytes transferred to the user;
0000 344 : and whether we update UCBSL_RECORD as a result of this HIC.
0000 345 :
0000 346 :
0000 347 :
0000 348 .MACRO TAT_ENTRY HIC, SOFTWARE_STATUS, -
0000 349 DEVDEPEND, ERRLOG, POSITION, COUNT, PREVTM, -
0000 350 SUBCODE_RUT
0000 351
0000 352 .BYTE HIC ; Hardware Interrupt Code
0000 353 .WORD SOFTWARE_STATUS ; Software status to return to user.
0000 354
0000 355 .IF NB DEVDEPEND
0000 356 .WORD DEVDEPEND@-16
0000 357 .IFF
0000 358 .WORD 0
0000 359 .ENDC
0000 360
0000 361 .IF IDENTICAL YES, <ERRLOG>
0000 362 ..XX..=TAT_M_ERRLOG
0000 363 .IFF
0000 364 ..XX..=0
0000 365 .ENDC
0000 366 .IF IDENTICAL YES, <POSITION>
0000 367 ..YY..=TAT_M_POSITION
0000 368 .IFF
0000 369 ..YY..=0
0000 370 .ENDC
0000 371 .IF IDENTICAL YES, <COUNT>
0000 372 ..ZZ..=TAT_M_COUNT
0000 373 .IFF
0000 374 ..ZZ..=0
0000 375 .ENDC
0000 376 .IF IDENTICAL YES, <PREVTM>
0000 377 ..QQ..=TAT_M_PREVTM
0000 378 .IFF
0000 379 ..QQ..=0
0000 380 .ENDC
0000 381
0000 382 .BYTE ..XX..!..YY..!..ZZ..!..QQ..
0000 383 .IF NB SUBCODE_RUT
0000 384 .LONG SUBCODE_RUT-TF$DDT
0000 385 .IFF
0000 386 .LONG 0
0000 387 .ENDC
0000 388 .ENDM
0000 389
0000 390 :
0000 391 : Expanded opcode macros - Branch word conditional psuedo opcodes.
0000 392 :
0000 393 :
0000 394 :
0000 395 : BUNEQ - Branch (word offset) not equal
0000 396 :
0000 397 :
0000 398 .MACRO BUNEQ DEST, ?L1
0000 399 BEQL L1 ; Branch around if NOT NEQ.
```

```
0000 400      BRW      DEST      ; Branch to destination if NEQ.
0000 401 L1:      ;          ; Around.
0000 402      .ENDM    BONEQ
0000 403
0000 404 ;
0000 405 ; BWEQL - Branch (word offset) equal
0000 406 ;
0000 407 ;
0000 408      .MACRO    BWEQL    DEST,?L1
0000 409      BNEQ      L1      ; Branch around if NOT EQL.
0000 410      BRW      DEST      ; Branch to destination if EQL.
0000 411 L1:      ;          ; Around.
0000 412      .ENDM    BWEQL
0000 413
0000 414 ;
0000 415 ; BWLSS - Branch (word offset) on less
0000 416 ;
0000 417 ;
0000 418      .MACRO    BWLSS    DEST,?L1
0000 419      BGEQ      L1      ; Branch around if NOT LSS.
0000 420      BRW      DEST      ; Branch to destination if LSS.
0000 421 L1:      ;          ; Around.
0000 422      .ENDM    BWLSS
0000 423
0000 424 ;
0000 425 ; BWGEQ - Branch (word offset) greater than or equal
0000 426 ;
0000 427 ;
0000 428      .MACRO    BWGEQ    DEST,?L1
0000 429      BLSS      L1      ; Branch around if NOT GEQ.
0000 430      BRW      DEST      ; Branch to destination if GEQ.
0000 431 L1:      ;          ; Around.
0000 432      .ENDM    BWGEQ
0000 433
0000 434 ;
0000 435 ; BWBS - Branch (word offset) bit set.
0000 436 ;
0000 437 ;
0000 438      .MACRO    BWBS      BIT,FIELD,DEST,?L1
0000 439      BBC      BIT,FIELD,L1 ; Branch around if bit NOT set.
0000 440      BRW      DEST      ; Branch to destination if bit set.
0000 441 L1:      ;          ; Around.
0000 442      .ENDM    BWBS
0000 443
0000 444 ;
0000 445 ; BWBC - Branch (word offset) bit clear.
0000 446 ;
0000 447 ;
0000 448      .MACRO    BWBC      BIT,FIELD,DEST,?L1
0000 449      BBS      BIT,FIELD,L1 ; Branch around if bit NOT clear.
0000 450      BRW      DEST      ; Branch to destination if bit clear.
0000 451 L1:      ;          ; Around.
0000 452      .ENDM    BWBC
```

```

0000 454 .SBTTL TM78/TU78 MASSBUS REGISTER OFFSETS
0000 455
0000 456 $DEFINI MF
0000 457
0000 458 $DEF MF_CS1 .BLKL 1 ;DRIVE CONTROL REGISTER
0004 459 _VIELD MF_CS1,0,<- ;DRIVE CONTROL REGISTER BIT DEFINITIONS
0004 460 <GO,,M>,- ;GO BIT
0004 461 <FCODE,5>,- ;FUNCTION CODE
0004 462 <IE,,M>,-
0004 463 <RDY,,M>,-
0004 464 <A16,,M>,-
0004 465 <A17,,M>,-
0004 466 <PSEL,,M>,-
0004 467 <DVA,,M>,- ;Device available
0004 468 <,1>,-
0004 469 <MCPÉ,,M>,-
0004 470 <TRE,,M>,-
0004 471 <SC,,M>,-
0004 472 >
0004 473 $DEF MF_IS .BLKL 1 ;INTERRUPT STATUS REGISTER.
0008 474 _VIELD MF_IS,0,<- ;Interrupt status dit definitions.
0008 475 <DTIC,6>,- ;Data transfer interrupt code.
0008 476 <,2>,- ;Reserved bits.
0008 477 <DPR,,M>,-
0008 478 <,1>,- ;Reserved bit.
0008 479 <DTFC,6>,- ;Data transfer failure code.
0008 480 >
0008 481 $DEF MF_TC .BLKL 1 ;TAPE CONTROL REGISTER.
000C 482 _VIELD MF_TC,0,<- ;Tape control bit definitions.
000C 483 <CMDADR,2>,- ;CMDADR specifies which tape unit.
000C 484 <RC,6,M>,- ;Record count.
000C 485 <SC,4>,- ;Skip count.
000C 486 <FMT,3>,- ;Format.
000C 487 <SER,,M>,- ;Suppress error repositioning.
000C 488 >
000C 489 $DEF MF_MR1 .BLKL 1 ;DIAGNOSTICS/MAINTENANCE REGISTER #1.
0010 490 $DEF MF_AB .BLKL 1 ;ATTENTION BIT REGISTER
0014 491 $DEF MF_BC .BLKL 1 ;BYTE COUNT REGISTER.
0018 492 $DEF MF_DT .BLKL 1 ;DRIVE TYPE REGISTER
001C 493 _VIELD MF_DT,0,<- ;Drive type register field definitions
001C 494 <DTN,9,M>,- ;Drive type number.
001C 495 <,1>,- ;Reserved bit.
001C 496 <,1>,- ;Reserved bit.
001C 497 <DMB,,M>,- ;Dual MASSBUS port option if set.
001C 498 <7CH,,M>,- ;7-Channel tape (always 0).
001C 499 <MOH,,M>,- ;Moving head (always 0).
001C 500 <TAP,,M>,- ;Tape drive (always 1).
001C 501 <NSA,,M>,- ;Not sector addressable (always 0).
001C 502 >
001C 503 $DEF MF_DS .BLKL 1 ;DRIVE STATUS REGISTER
0020 504 _VIELD MF_DS,0,<- ;Drive status register bit definitions.
0020 505 <,4>,- ;Reserved bits.
0020 506 <DSE,,M>,- ;Unit doing erase portion of DSE cmd.
0020 507 <MAINT,,M>,- ;Unit is in maintenance mode.
0020 508 <SHR,,M>,- ;Unit available to both MASSBUS ports.
0020 509 <AVAIL,,M>,- ;Unit available to this MASSBUS.
0020 510 <FPT,,M>,- ;Unit is file protected.

```

0020	511		<EOT,,M>,-			: End of tape.
0020	512		<BOT,,M>,-			: Beginning of tape.
0020	513		<PE,,M>,-			: Unit set for PE (1600 BPI) format.
0020	514		<REW,,M>,-			: Rewind is in progress.
0020	515		<ONL,,M>,-			: Unit on-line, tape is mounted.
0020	516		<PRES,,M>,-			: Unit is powered up.
0020	517		<RDY,,M>,-			: Unit ready.
0020	518		>			
0020	519	\$DEF	MF_SN	.BLKL	1	: SERIAL NUMBER REGISTER
0024	520	\$DEF	MF_MR2	.BLKL	1	: DIAGNOSTICS/MAINTENANCE REGISTER #2.
0028	521	\$DEF	MF_MR3	.BLKL	1	: DIAGNOSTICS/MAINTENANCE REGISTER #3.
002C	522	\$DEF	MF_NDTA	.BLKL	1	: NON-DATA TRANSFER ATTENTION REGISTER.
0030	523		_VIELD MF_NDTA,0,<-			: Non-data transfer field definitions.
0030	524		<NDIC,6>,-			: NDT interrupt code.
0030	525		<,2>,-			: Reserved bits.
0030	526		<ATAD,2>,-			: Attention address (unit #).
0030	527		<NDFC,6>,-			: NDT failure code.
0030	528		>			
0030	529	\$DEF	MF_NDT0	.BLKL	1	: NONDATA TRANSFER CMD REG. (UNIT #0).
0034	530		_VIELD MF_NDT0,0,<-			: Nondata transfer command field defs.
0034	531		<GO,,M>,-			: Go bit.
0034	532		<FCODE,5>,-			: Function code.
0034	533		<,2>,-			: Reserved bits.
0034	534		<CCNT,8>,-			: Command count.
0034	535		>			
0034	536	\$DEF	MF_NDT1	.BLKL	1	: NONDATA TRANSFER CMD REG. (UNIT #1).
0038	537		_VIELD MF_NDT1,0,<-			: Nondata transfer command field defs.
0038	538		<GO,,M>,-			: Go bit.
0038	539		<FCODE,5>,-			: Function code.
0038	540		<,2>,-			: Reserved bits.
0038	541		<CCNT,8>,-			: Command count.
0038	542		>			
0038	543	\$DEF	MF_NDT2	.BLKL	1	: NONDATA TRANSFER CMD REG. (UNIT #2).
003C	544		_VIELD MF_NDT2,0,<-			: Nondata transfer command field defs.
003C	545		<GO,,M>,-			: Go bit.
003C	546		<FCODE,5>,-			: Function code.
003C	547		<,2>,-			: Reserved bits.
003C	548		<CCNT,8>,-			: Command count.
003C	549		>			
003C	550	\$DEF	MF_NDT3	.BLKL	1	: NONDATA TRANSFER CMD REG. (UNIT #3).
0040	551		_VIELD MF_NDT3,0,<-			: Nondata transfer command field defs.
0040	552		<GO,,M>,-			: Go bit.
0040	553		<FCODE,5>,-			: Function code.
0040	554		<,2>,-			: Reserved bits.
0040	555		<CCNT,8>,-			: Command count.
0040	556		>			
0040	557	\$DEF	MF_IA	.BLKL	1	: INTERNAL ADDRESS REGISTER.
0044	558	\$DEF	MF_ID	.BLKL	1	: INTERNAL DATA REGISTER.
0048	559		_VIELD MF_ID,0,<-			: Internal data field definitions.
0048	560		<ID,8>,-			: Actual internal data to be sent.
0048	561		<HOLD,,M>,-			
0048	562		<HLDA,,M>,-			: (Read-only)
0048	563		<EVPAR,,M>,-			
0048	564		<CPE,,M>,-			: (Read only)
0048	565		<ILR,,M>,-			: (Read only)
0048	566		<MCPE,,M>,-			: (Read only)
0048	567		<TMCLR,,M>,-			

TFDRIVER
V04-000

F5

16-SEP-1984 00:08:15 VAX/VMS Macro V04-00
5-SEP-1984 00:17:37 [DRIVER.SRC]TFDRIVER.MAR;1

Page 11
(1)

- TM78/TU78 MAGTAPE DRIVER
TM78/TU78 MASSBUS REGISTER OFFSETS

0048 568
0048 569
0048 570
0048 571

<TMRDY,,M>,-
>
\$DEFEND MF

TFI
VO

```
0000 573 .SBTTL TM78/TU78 DEVICE DEPENDENT UNIT CONTROL BLOCK OFFSETS
0000 574
0000000F 0000 575 MAX_SUBSTACK_DEPTH=15
0000 576
0000 577 $DEFINI UCB
0000 578
0000 579 $VIELD UCB,0,<-
0000 580 <MF_REWIND,,M>,- ;DEVICE DEPENDENT STATUS BITS
0000 581 <MF_POWER,,M>,- ; REWIND IN PROGRESS
0000 582 <MF_CNCLP,,M>,- ; Currently executing in POWERFAIL_TIMEOUT
0000 583 <MF_REPOS,,M>,- ; CANCEL I/O pending.
0000 584 <MF_TMRDY,,M>,- ; Tape needs repositioning.
0000 585 <MF_ATTN,,M>,- ; TM CLR has been issued; TMRDY pending
0000 586 - ; 'B' type fault has occurred; issue
0000 587 <MF_UCBFREE,,M>,- ; TMCLR at earliest moment.
0000 588 <MF_UCBBUSY,,M>,- ; UCB may be used for ASYNCHRONOUS fork.
0000 589 <MF_SOFT_ERR,,M>,- ; UCB being used for ASYNCHRONOUS fork.
0000 590 <MF_EXSNS_DONE,,M>,- ; Pass SOFT ERROR flag to DEVICE_ERROR.
0000 591 <MF_OWNPCHN,,M>,- ; Extended sense already done.
0000 592 <MF_OWNSCHN,,M>,- ;
0000 593 > ; Bits whose value are only of interest
0C00 594 ; to the routine that invokes EXTENDED
0000 595 ; SENSE command. Since this command
0000 596 ; requires ownership of the channels,
0000 597 ; these bits indicate whether the
0000 598 ; channels were owned prior to calling
0000 599 ; the routine. Therefore the
0000 600 ; MF_OWNPCHN bit set means that the
0000 601 ; primary (i.e. the TM78) channel was
0000 602 ; owned by the caller of the routine
0000 603 ; and the MF_OWNSCHN bit set means that
0000 604 ; the secondary (i.e. MBA) channel was
0000 605 ; owned. Previously owned channels are
0000 606 ; not released prior to return.
0000 607 ; NOTE- TM CLR resets TM78 and is only used
0000 608 ; if a FAULT B or an MBA error has occurred
0000 609 ; NOTE - The UCBSM_MF_UCBFREE and UCBSM_MF_UCBBUSY bits in UCBSW_DEVSTS are
0000 610 ; interpreted in conjunction with the UCBSM_BSY bit in UCBSW_STS. The
0000 611 ; UCBSM_MF_UCBFREE bit equal to 1 (i.e. set) implies that no one is currently
0000 612 ; executing in the context of the UCB between STARTIO and FUNCTION_EXIT.
0000 613 ; The bit is originally set (to 1) at system init time, reset upon entry
0000 614 ; to STARTIO and then set again in FUNCTION_EXIT prior to REQCOM. When
0000 615 ; set (to 1) it declares to the Interrupt routine that the Interrupt
0000 616 ; routine CAN do an IOFORK and a call to ISSUE TMCLR since the UCB is
0000 617 ; in a state that is guaranteed to not do another IOFORK on top of this
0000 618 ; one. When UCBSM_MF_UCBFREE is zero, it declares that an IOFORK may not be
0000 619 ; issued asynchronously and the Interrupt routine must instead simply
0000 620 ; set the UCBSM_MF_ATTN in the event of a 'B' type fault attention.
0000 621 ;
0000 622 ; UCBSM_MF_UCBBUSY in UCBSW_DEVSTS is turned on (to 1) by the Interrupt
0000 623 ; routine when in servicing a 'B' type fault, it finds UCBSM_MF_UCBFREE set and
0000 624 ; therefore decides to take over the fork block in the UCB by issuing
0000 625 ; an IOFORK. It sets UCBSM_MF_UCBBUSY to warn a driver process entering STARTIO
0000 626 ; of the situation. Such a driver process, upon finding the UCBSM_MF_UCBBUSY bit
0000 627 ; on, would simply execute an RSB instruction. The I/O function will
0000 628 ; be started later when the TMCLR is resolved.
0000 629 ;
```

TFDRIVER
V04-000

- TM78/TU78 MAGTAPE DRIVER
TM78/TU78 DEVICE DEPENDENT UNIT CONTROL

H 5

16-SEP-1984 00:08:15
5-SEP-1984 00:17:37

VAX/VMS Macro V04-00
[DRIVER.SRC]TFDRIVER.MAR;1

Page 13
(1)

TF
V0

0000 630


```
000000B4 0C30 632
0000 633 . =UCBSK_LCL_TAPE_LENGTH
00B4 634
00B4 635 $DEF UCBSW_MF_CS1 .BLKW 1 ;Setup space for Control register.
00B6 636 $DEF UCBSW_MF_TC .BLKW 1 ;Saved Tapedrive Control register.
00B8 637 $DEF UCBSW_MF_TMPLTC .BLKW 1 ;Template Tapedrive Control register.
00BA 638 $DEF UCBSB_MF_DENSITY
000000BB 00BA 639 .BLKB 1 ;Current DENSITY of drive.
00BB 640 ; 0=> 1600 BPI - 1 => 6250 BPI
00BB 641 $DEF UCBSB_MF_RSTCNT .BLKB 1 ; Count of number of times this
00BC 642 ; I/O function has been restarted.
00BC 643
00BC 644 $DEF UCBSW_MF_NDTCR .BLKW 1 ; Non-data transfer command word.
00BE 645 ; Scratch space where we build a
00BE 646 ; non-data transfer command and count.
00BE 647 $DEF UCBSW_MF_MAX_REWIND
000000C0 00BE 648 .BLKW 1 ; Maximum time that a rewind can take
00C0 649 ; for this unit before a timeout.
00C0 650 $DEF UCBSL_MF_TIMEOUT .BLKL 1 ; Number of seconds for TIMEOUT
00C4 651 ; parameter to WFIKPC macro.
00C4 652
00C4 653 $DEF UCBSL_MF_ORGPOS .BLKL 1 ; Original value of UCBSL_RECORD at
00C8 654 ; start of current I/O operation.
00C8 655 ; Used in POWERFAIL recovery.
00C8 656
00C8 657 $DEF UCBSL_MF_PREVTM .BLKL 1 ; Position of previous TAPEMARK; used
00CC 658 ; in forward SKIPFILE and SPACEFILE
00CC 659 ; operations in detecting consecutive
00CC 660 ; TAPEMARKS.
00CC 661
00CC 662 $DEF UCBSL_MF_ORGPTM .BLKL 1 ; Original value of UCBSL_MF_PREVTM at
00D0 663 ; start of current I/O operation.
00D0 664 ; Used in POWERFAIL recovery.
00D0 665
00D0 666 $DEF UCBSL_MF_SVAPTE .BLKL 1 ; Virtual address of the System PTE for
00D4 667 ; UCBSL_MF_EXSNS.
00D4 668 $DEF UCBSW_MF_BOFF .BLKW 1 ; Byte offset of UCBSL_MF_EXSNS in its page.
00D6 669 $DEF UCBSW_MF_BCNT .BLKW 1 ; Count of # of bytes (60) transferred
00D8 670 ; by an EXTENDED SENSE command.
00D8 671 $DEF UCBSQ_MF_TEMP .BLKL 2 ; Temporary space used to store UCBSL_SVAPTE
00E0 672 ; and UCBSW_BOFF during EXTENDED SENSE.
00E0 673
00E0 674 $DEF UCBSL_MF_SUBSP .BLKL 1 ; SUBSTACK depth indicator (Stack ptr).
00E4 675
00E4 676 $DEF UCBSL_MF_SUBSTACK ; Space for SUBSTACK array.
00000120 00E4 677 .BLKL MAX_SUBSTACK_DEPTH
0120 678
0120 679 ;
0120 680 ; Space to save device registers for error logging.
0120 681 ;
0120 682
0120 683 $DEF UCBSL_MFMBA_CSR .BLKL 1 ;Saved MBA Configuration register.
0124 684 $DEF UCBSL_MFMBA_CR .BLKL 1 ;Saved MBA Control register.
0128 685 $DEF UCBSL_MFMBA_SR .BLKL 1 ;Saved MBA Status register.
012C 686 $DEF UCBSL_MFMBA_VAR .BLKL 1 ;Saved MBA Virtual Address register.
0130 687 $DEF UCBSL_MFMBA_BCR .BLKL 1 ;Saved MBA Byte Count Register.
0134 688 $DEF UCBSL_MFMBA_FMAP
```

```
00000138 0134 689 ;Saved MBA Final Map register.
0000013C 0138 690 $DEF UCB$$_MF MBA_PMAP .BLKL 1 ;Saved MBA Previous Map register.
013C 691
013C 692
013C 693 $DEF UCB$$_MF_CS1 .BLKL 1 ;Saved data transfer Control register.
0140 694 $DEF UCB$$_MF_IS .BLKL 1 ;Saved data transfer Interrupt status.
0144 695 $DEF UCB$$_MF_TC .BLKL 1 ;Saved data transfer Tape control.
0148 696 $DEF UCB$$_MF_MR1 .BLKL 1 ;Saved Maintenance Register 1.
014C 697 $DEF UCB$$_MF_AB .BLKL 1 ;Saved Attention Bit.
0150 698 $DEF UCB$$_MF_BC .BLKL 1 ;Saved data transfer Byte Count reg.
0154 699 $DEF UCB$$_MF_DT .BLKL 1 ;Saved Device Type register.
0158 700 $DEF UCB$$_MF_DS .BLKL 1 ;Saved Drive Status register.
015C 701 $DEF UCB$$_MF_SN .BLKL 1 ;Saved Serial Number register.
0160 702 $DEF UCB$$_MF_MR2 .BLKL 1 ;saved Maintenance Register 2.
0164 703 $DEF UCB$$_MF_MR3 .BLKL 1 ;saved Maintenance Register 3.
0168 704 $DEF UCB$$_MF_NDTA .BLKL 1 ;Saved Non-Data Transfer Attention reg.
016C 705 $DEF UCB$$_MF_NDT0 .BLKL 1 ;Saved NDT command register drive 0.
0170 706 $DEF UCB$$_MF_NDT1 .BLKL 1 ;Saved NDT command register drive 1.
0174 707 $DEF UCB$$_MF_NDT2 .BLKL 1 ;Saved NDT command register drive 2.
0178 708 $DEF UCB$$_MF_NDT3 .BLKL 1 ;Saved NDT command register drive 3.
017C 709 $DEF UCB$$_MF_ID .BLKL 1 ;Saved Internal Data register.
0180 710
00000044 0180 711 UCB_TM78REGS_LEN=-UCB$$_MF_CS1 ; Length to store all TM78 registers.
0180 712
0180 713 $DEF UCB$$_MF_CMD .BLKL 1 ;Copy of currently active TM78 COMMAND
0184 714 ; (for this UCB) be it NDT or Transfer.
0184 715 $DEF UCB$$_MF_NDTA_C .BLKL 1 ; Space to copy contents of UCB$$_MF_NDTA
0188 716 ; immediately after wakeup from WFI RCH
0188 717 ; so that we can preserve a stable copy
0188 718 ; of the attention data associated with
0188 719 ; the interrupt.
0188 720
0188 721 $DEF UCB$$_MF_EXSNS .BLKL 15 ; Space for output of EXTENDED SENSE.
01C4 722
000000A4 01C4 723 UCB_REGDUMP_LEN=-UCB$$_MF MBA_CSR ; Length of data to copy to ERROR LOG.
01C4 724
01C4 725
000001C4 01C4 726 UCB$$_MF_LENGTH= ; Length of a TU78 UCB.
01C4 727
01C4 728 $DEFEND UCB
0000 729
0000 730 ;
0000 731 ; Maximum rewind times for the units that can be on a TM78. For now these
0000 732 ; are only TU78's.
0000 733 ;
00000096 0000 734 TU78_MAX_REWIND=150 ; (3600*12)/440 + slop factor.
000002A3 0000 735 TU78_MAX_DSE=675 ; Approximately 4.5*TU78_MAX_REWIND
0000020D 0000 736 TU78_MAX_SPACE=525 ; In bizarre situations a tape could have huge
0000 737 ; physical records which could mean that the
0000 738 ; entire tape might contain less than
0000 739 ; 255 records so that the maximum spacing time
0000 740 ; must be the time to space the entire tape at
0000 741 ; 125 IPS. Approximately 3.5 * TU78_MAX_REWIND
00000005 0000 742 MAX_RESTARTIO=5 ; Maximum number of times to allow RESTARTIO to
0000 743 ; be attempted before giving up on the I/O
0000 744 ; operation.
00000000 0000 745 MF$$_DENSITY_1600=0 ; Value put in UCB$$_MF_DENSITY for 1600 BPI.
```

TFDRIVER
V04-000

K 5
- TM78/TU78 MAGTAPE DRIVER
TM78/TU78 DEVICE DEPENDENT UNIT CONTROL

16-SEP-1984 00:08:15 VAX/VMS Macro V04-00 Page 16
5-SEP-1984 00:17:37 [DRIVER.SRC]TFDRIVER.MAR;1 (1)

TF
V0

00000001	0000	746	MFSK DENSITY 6250=1	:	Value in UCB\$B MF DENSITY which means 6250 BPI
00000005	0000	747	MINIMUM_TIMEOUT=5	:	Generous minimum timeout for TU78 operations.
	0000	748		:	All data transfer operations and most Non-data
	0000	749		:	transfer operations specify this amount.
6D29504E	0000	750	MEDIA_ID_TU78 = ^X<6D29504E>	:	Media id type for the TU78.

```
00000001 0000 752 .SBTTL Hardware function and interrupt codes
00000001 0000 753 GO_BIT=1
00000001 0000 754
00000001 0000 755 ; Non-data transfer hardware function codes.
00000001 0000 756
00000002 0000 757 F_NOP=1*2 ; No-op
00000004 0000 758 F_UNLOAD=2*2 ; Unload tape and interrupt immediately
00000006 0000 759 F_REWIND=3*2 ; Rewind tape and interrupt after tape motion stops
00000008 0000 760 F_SENSE=4*2 ; Put status info into MF_DT, MF_DS and MF_SN registers
0000000A 0000 761 ; NOTE- Status valid only while attention bit is set
0000000A 0000 762 F_DSE=5*2 ; Erase remainder of tape and rewind
0000000C 0000 763 ; NOTE- Erases at least 10 feet beyond EOT marker
0000000C 0000 764 F_WTM_PE=6*2 ; Write PE (1600 BPI) tape mark. NOTE- Recording format
0000000C 0000 765 ; is ignored except when tape is at load point (BOT).
0000000C 0000 766 ; It is specified by the low-order bit of code.
0000000E 0000 767 F_WTM_GCR=7*2 ; Write GCR (6250 BPI) tape mark. NOTE- Recording format
0000000E 0000 768 ; is ignored except when tape is at load point (BOT).
0000000E 0000 769 ; It is specified by the low-order bit of code.
00000010 0000 770 F_SP_FWD_REC=8*2 ; Space forward record, stop if tape mark
00000012 0000 771 F_SP_REV_REC=9*2 ; Space reverse record, stop if tape mark or BOT
00000014 0000 772 F_SP_FWD_FILE=10*2 ; Space forward file (to tape mark)
00000016 0000 773 F_SP_REV_FILE=11*2 ; Space reverse file (to tape mark)
00000018 0000 774 F_SP_FWD_EITHER=12*2 ; Space forward either record or file
0000001A 0000 775 F_SP_REV_EITHER=13*2 ; Space reverse either record or file
0000001C 0000 776 F_ERG_PE=14*2 ; Erase three inches of tape, set PE (1600).
0000001C 0000 777 ; NOTE- Recording format is ignored except at BOT.
0000001C 0000 778 ; It is specified by the low-order bit of code.
0000001E 0000 779 F_ERG_GCR=15*2 ; Erase three inches of tape, set GCR (6250).
0000001E 0000 780 ; NOTE- Recording format is ignored except at BOT.
0000001E 0000 781 ; It is specified by the low-order bit of code.
00000020 0000 782 F_CLOSE_FILE_PE=16*2 ; Write 2 tape marks, space reverse 1, set PE.
00000020 0000 783 ; NOTE- Recording format is ignored except at BOT.
00000020 0000 784 ; It is specified by the low-order bit of code.
00000022 0000 785 F_CLOSE_FILE_GCR=17*2 ; Write 2 tape marks, space reverse 1, set GCR.
00000022 0000 786 ; NOTE- Recording format is ignored except at BOT.
00000022 0000 787 ; It is specified by the low-order bit of code.
00000024 0000 788 F_SP_LEOT=18*2 ; Space forward until 2 tape marks, space reverse 1
00000026 0000 789 F_SP_FWD_FILE_LEOT=19*2 ; Space forward to tape mark, stop if 2 successive
00000026 0000 790 ; tape marks (LEOT), then space reverse one tape mark.
00000026 0000 791 ; NOTE- Not to be used after any reverse operation,
00000026 0000 792 ; or the TM78 may skip over an LEOT located
00000026 0000 793 ; where direction was reversed.
00000026 0000 794
00000026 0000 795 ; Data transfer hardware function codes
00000026 0000 796
00000028 0000 797 F_WRITE_CHECK_FWD=20*2 ; Write Check Forward. Tape drive reads a record in the
00000028 0000 798 ; forward direction while at the same time the MBA
00000028 0000 799 ; reads a buffer from memory and compares the buffer
00000028 0000 800 ; to the contents of the record read from tape.
00000028 0000 801
0000002E 0000 802 F_WRITE_CHECK_REV=23*2 ; Write Check Reverse. Same as previous except the
0000002E 0000 803 ; record is read in the reverse direction and the
0000002E 0000 804 ; buffer contents are compared in the reverse
0000002E 0000 805 ; direction also.
0000002E 0000 806
00000030 0000 807 F_WRITE_PE=24*2 ; Write PE (1600 BPI) encoded records. -NOTE- The
00000030 0000 808 ; recording format is ignored unless the tape is posi-
```

TFDRIVER
V04-000

M 5
- TM78/TU78 MAGTAPE DRIVER
Hardware function and interrupt codes

16-SEP-1984 00:08:15 VAX/VMS Macro V04-00 Page 18
5-SEP-1984 00:17:37 [DRIVER.SRC]TFDRIVER.MAR;1 (1)

	0000	809
	0000	810
00000032	0000	811 F_WRITE_GCR=25*2
	0000	812
	0000	813
	0000	814
00000038	0000	815 F_READ_FWD=28*2
0000003E	0000	816 F_READ_REV=31*2
0000003A	0000	817 F_EXSNS=29*2

; tioned at the load point. At load point, the write
; command specifies the recording format of the entire tap
; Write GCR (6250 BPI) encoded records. -NOTE- The
; recording format is ignored unless the tape is posi-
; tioned at the load point. At load point, the write
; command specifies the recording format of the entire tap
; Read records forward
; Read records reverse
; Extended sense

```
0000 819 ; HARDWARE INTERRUPT CODES
0000 820
0000 821 ;
0000 822 ; Interrupt codes that may result from both data transfer operations and
0000 823 ; non-data transfer operations.
0000 824 ;
0000 825
00000001 0000 826 HIC_DONE=1 ; Operation completed as expected
00000002 0000 827 HIC_TM=2 ; Unexpected tape mark
00000003 0000 828 HIC_BOT=3 ; Unexpected beginning of tape
00000004 0000 829 HIC_EOT=4 ; Tape is positioned beyond end of tape marker (write)
00000008 0000 830 HIC_FPT=8 ; Write was attempted on a file protected tape
00000009 0000 831 HIC_NOT_RDY=9 ; TU is on line but tape is rewinding or loading, or
0000 832 ; a DSE or rewind is being performed from the other
0000 833 ; port
0000000A 0000 834 HIC_NOT_AVAIL=10 ; TU is not switched to this port, but is on-line
0000000B 0000 835 HIC_OFF_LINE=11 ; TU is not switched on-line with a tape loaded
0000000C 0000 836 HIC_NON_EX=12 ; TU does not exist or power is off
0000000D 0000 837 HIC_NOT_CAPABLE=13 ; 1. Incorrect I.D. burst
0000 838 ; 2. No record or tape mark detected
00000017 0000 839 HIC_BAD_TAPE=23 ; Tape position is lost
00000018 0000 840 HIC_TM_FAULT_A=24 ; Software (microcode) or TM78 hardware is broken -
0000 841 ; see failure code for details
00000019 0000 842 HIC_TU_FAULT_A=25 ; Tape unit hardware is broken - see failure code
0000 843
0000 844 ;
0000 845 ; Interrupt codes which result only from data transfer operations.
0000 846 ;
0000 847
00000010 0000 848 HIC_LONG_REC=16 ; Last record read was longer than BYTE COUNT value,
0000 849 ; but was otherwise correctly read. The tape position
0000 850 ; is after the long record. BYTE COUNT is set to actual
0000 851 ; record length. RECORD COUNT is set to number of
0000 852 ; records left.
0000 853
00000011 0000 854 HIC_SHORT_REC=17 ; Last record read was shorter than initial BYTE COUNT
0000 855 ; value, but is otherwise correctly in memory. Tape
0000 856 ; position is after the short record. BYTE COUNT is
0000 857 ; set to the actual record length. RECORD COUNT is
0000 858 ; set to the number of records left.
0000 859
00000012 0000 860 HIC_RETRY=18 ; Error, the initial operation should be repeated.
0000 861 ; Tape positioned for retry. BYTE COUNT is set to
0000 862 ; its initial value. RECORD COUNT is set to the number
0000 863 ; of records left.
0000 864
00000013 0000 865 HIC_READ_OPP=19 ; Read error. The initial read should be performed in
0000 866 ; the opposite direction. Tape is positioned after the
0000 867 ; bad record. BYTE COUNT is set to number bytes to
0000 868 ; be read. RECORD COUNT is set to number of records left.
0000 869
00000014 0000 870 HIC_UNREADABLE=20 ; Read retries have failed to read the record. Tape is
0000 871 ; positioned after the record. BYTE COUNT is set to
0000 872 ; the number of bytes transferred. RECORD COUNT is set
0000 873 ; to the number of records left.
0000 874
00000015 0000 875 HIC_ERROR=21 ; An error has occurred which requires a retry, but SER
```

```
0000 876 ; is set. (EXCLUDES length errors). Tape is positioned
0000 877 ; after the bad record. BYTE COUNT is set to the number
0000 878 ; of bytes transferred. RECORD COUNT is set to the number
0000 879 ; of records left.
0000 880
00000016 0000 881 HIC_EOT_ERROR=22 ; A write error has occurred beyond the EOT marker, and
0000 882 ; SER is set. Tape is positioned after the bad record.
0000 883 ; BYTE COUNT is set to the number of records transferred.
0000 884 ; RECORD COUNT is set to the number of records left.
0000 885
0000 886 ;
0000 887 ; Interrupt codes which result only from non-data transfer operations.
0000 888 ;
0000 889
00000005 0000 890 HIC_LEOT=5 ; Unexpected logical EOT (2 tape marks)
00000006 0000 891 HIC_NOP=6 ; No-op completed
00000007 0000 892 HIC_REWINDING=7 ; Rewind in progress, done interrupt will follow when
0000 893 ; at BOT
0000 894
0000 895 ;
0000 896 ; TM78 initiated interrupt codes.
0000 897 ;
0000 898
0000000F 0000 899 HIC_ON_LINE=15 ; A tape unit has come on-line with a tape loaded, or
0000 900 ; a rewind has finished on the other port for the
0000 901 ; accessed tape unit.
0000001A 0000 902 HIC_TM_FAULT_B=26 ; The TM78 is broken.
0000001B 0000 903 HIC_TU_FAULT_B=27 ; The tape unit is broken.
0000001C 0000 904 HIC_MB_FAULT=28 ; A MASSBUS error has occurred.
0000000E 0000 905 HFC_EOT=14 ; Failure code means drive is positioned at or beynd EOT
0000 906
0000 907 ;
0000 908 ; TM78 revision codes
0000 909 ;
00000006 0000 910 REV_LOC_0=*0<06>
00000005 0000 911 REV_LOC_1=*0<05>
00000006 0000 912 REV_LOC_2=*0<06>
00000004 0000 913 REV_LOC_3=*0<04>
00000002 0000 914 REV_LOC_4=*0<02>
00000003 0000 915 REV_LOC_5=*0<03>
00000008 0000 916 REV_LOC_6=*0<10>
00000003 0000 917 REV_LOC_7=*0<03>
0000 918
```

```
0000 920 :  
0000 921 : LOCAL DATA  
0000 922 :  
0000 923 : DRIVER PROLOGUE TABLE  
0000 924 :  
0000 925 :  
0000 926 DPTAB - ;DEFINE DRIVER PROLOGUE TABLE  
0000 927 END=TF_END,- ;END OF DRIVER  
0000 928 FLAGS=DPT$M_SUBCNTRL,- ;INDICATE SUBCONTROLLER  
0000 929 ADAPTER=MBA,- ;ADAPTER TYPE  
0000 930 UCBSIZE=UCB$K_MF_LENGTH,- ;UCB SIZE  
0000 931 MAXUNITS=4,- ;TM78 LIMITED TO 4 TU78'S  
0000 932 NAME=TFDRIVER ;DRIVER NAME  
0038 933 DPT_STORE INIT ;CONTROL BLOCK INIT VALUES  
0038 934 DPT_STORE DDB, DDB$$_ACPD, L, <^A\M\TA> ;DEFAULT ACP NAME  
003F 935 DPT_STORE UCB, UCB$$_FIPL, B, 8 ;FORK IPL  
0043 936 DPT_STORE UCB, UCB$$_DEVCHAR, L, - ;DEVICE CHARACTERISTICS  
0043 937 <DEV$M_FOD- ;FILES ORIENTED  
0043 938 :DEV$M_DIR- ;DIRECTORY STRUCTURED  
0043 939 :DEV$M_AVL- ;AVAILABLE  
0043 940 :DEV$M_ELG- ;ERROR LOGGING ENABLED  
0043 941 :DEV$M_IDV- ;INPUT DEVICE  
0043 942 :DEV$M_ODV- ;OUTPUT DEVICE  
0043 943 :DEV$M_SDI- ;SINGLE DIRECTORY DEVICE  
0043 944 :DEV$M_SQD> ;SEQUENTIAL DEV. CE  
004A 945 DPT_STORE UCB, UCB$$_DEVCHAR2, L, - ;DEVICE CHARACTERISTICS  
004A 946 <DEV$M_NNM> ;PREFIX NAME WITH 'node$'  
0051 947 DPT_STORE UCB, UCB$$_DEVCLASS, B, DCS_TAPE ;DEVICE CLASS  
0055 948 DPT_STORE UCB, UCB$$_DEVBUFSIZ, W, 2048 ;DEFAULT BUFFER SIZE  
005A 949 DPT_STORE UCB, UCB$$_DEVDEPEND, W, <^X4C0> ;DEFAULT TAPE PARAMETERS  
005F 950 DPT_STORE UCB, UCB$$_DIPL, B, 21 ;DEVICE IPL  
0063 951 DPT_STORE UCB, UCB$$_ERTCNT, B, 127 ;Error retry count  
0067 952 DPT_STORE UCB, UCB$$_ERTMAX, B, 127 ;Maximum error retry  
006B 953 DPT_STORE REINIT ;CONTROL BLOCK RE-INIT VALUES  
006B 954 DPT_STORE CRB, CRB$$_INTD+4, D, TFS$INT ;INTERRUPT SERVICE ROUTINE ADDRESS  
0070 955 DPT_STORE CRB, CRB$$_INTD+VEC$$_INITIAL, D, TM78_INIT ;CONTROLLER INIT  
0075 956 DPT_STORE CRB, CRB$$_INTD+VEC$$_UNITINIT, D, TM78_TXXX_INIT ;UNIT INIT  
007A 957 DPT_STORE DDB, DDB$$_DDT, D, TFS$DDT ;DDT ADDRESS  
007F 958 DPT_STORE END ;  
00000001 0000 959 .MDELETE DPT_STORE  
0000 960  
0000 961 :  
0000 962 : DRIVER DISPATCH TABLE  
0000 963 :  
0000 964 :  
0000 965 DDTAB TF,- ;DRIVER DISPATCH TABLE  
0000 966 TF_STARTIO,- ;START I/O OPERATION  
0000 967 TF_UNSLNT,- ;UNSOLICITED INTERRUPT  
0000 968 TF_FUNCTABLE,- ;FUNCTION DECISION TABLE  
0000 969 TF_CANCELIO,- ;CANCEL I/O ENTRY POINT  
0000 970 TF_REGDUMP,- ;REGISTER DUMP ROUTINE  
0000 971 <<UCB_REGDUMP_LEN>+<<3+5+1>*4>>,- ;DIAGNOSTIC BUFFER SIZE  
0000 972 <<UCB_REGDUMP_LEN>+<1+4>+<EMBSL_DV_REGSAV>> ;ERROR BUFFER SIZE  
0038 973  
0038 974 :  
0038 975 : TM78 DRIVE TYPE DESCRIPTOR TABLE  
0038 976 :
```



```
0038 977
0038 978 TF_DTDESC:
0041 0038 979 .WORD ^X41 ;TU78 125 IPS
00 003A 980 .BYTE DT$ TU78
00000003 003B 981 TF_DTDESCLEN=-TF_DTDESC ;LENGTH OF DRIVE TYPE DESCRIPTOR
0000 003B 982 .WORD 0 ;END OF TABLE
00000040 003D 983 .BLKB TF_DTDESCLEN ;SPARE DRIVE TYPE SLOT
0040 984
0040 985 ;
0040 986 ; Format code translation table
0040 987 ;
0040 988 ; The following table relates software defined tape formats to
0040 989 ; TM78 specific codes used to realize these formats. The table
0040 990 ; contains 16 bytes, one for each of the possible values of the
0040 991 ; 4 bit field which specifies the software format (MTSM_FORMAT).
0040 992 ; The contents of each byte of the table is the respective code
0040 993 ; to load into the TM78 register.
0040 994 ;
0040 995 ;
0040 996 ASSUME MT$S_FORMAT EQ 4 ; Assume a 4 bit field.
0040 997 ASSUME MT$K_DEFAULT EQ 0 ; Assume values defined by STARDEF.MDL
0040 998 ASSUME MT$K_NORMAL11 EQ 12
0040 999 ASSUME MT$K_CORDMP11 EQ 13
0040 1000 ASSUME MT$K_NORMAL15 EQ 14
0040 1001
00000000 0040 1002 MFSK_FORMAT_NORMAL_11=0 ; TM78 code to specify normal 11 tape format.
00000001 0040 1003 MFSK_FORMAT_NORMAL_15=1 ; TM78 code to specify normal 15 tape format.
00000002 0040 1004 MFSK_FORMAT_COMPATIBLE_10=2 ; TM78 code to specify PDP 10 tape format.
00000003 0040 1005 MFSK_FORMAT_COREDUMP_10=3 ; TM78 code to specify PDP 10 coredump format.
00000004 0040 1006 MFSK_FORMAT_HI_DENS_COMPAT_10=4 ; TM78 code to specify high density PDP 10 format.
00000005 0040 1007 MFSK_FORMAT_IMAGE=5 ; TM78 code to specify IMAGE tape format.
00000006 0040 1008 MFSK_FORMAT_HI_DENS_DUMP_10=6 ; TM78 code to specify high density PDP 10 dump form
0040 1009 FORMAT_TABLE:
00 0040 1010 .BYTE MFSK_FORMAT_NORMAL_11 ; Defaults to NORMAL11.
00 0041 1011 .BYTE MFSK_FORMAT_NORMAL_11 ; Illegal value defaults to NORMAL11.
00 0042 1012 .BYTE MFSK_FORMAT_NORMAL_11 ; Illegal value defaults to NORMAL11.
00 0043 1013 .BYTE MFSK_FORMAT_NORMAL_11 ; Illegal value defaults to NORMAL11.
00 0044 1014 .BYTE MFSK_FORMAT_NORMAL_11 ; Illegal value defaults to NORMAL11.
00 0045 1015 .BYTE MFSK_FORMAT_NORMAL_11 ; Illegal value defaults to NORMAL11.
00 0046 1016 .BYTE MFSK_FORMAT_NORMAL_11 ; Illegal value defaults to NORMAL11.
00 0047 1017 .BYTE MFSK_FORMAT_NORMAL_11 ; Illegal value defaults to NORMAL11.
00 0048 1018 .BYTE MFSK_FORMAT_NORMAL_11 ; Illegal value defaults to NORMAL11.
00 0049 1019 .BYTE MFSK_FORMAT_NORMAL_11 ; Illegal value defaults to NORMAL11.
00 004A 1020 .BYTE MFSK_FORMAT_NORMAL_11 ; Illegal value defaults to NORMAL11.
00 004B 1021 .BYTE MFSK_FORMAT_NORMAL_11 ; Illegal value defaults to NORMAL11.
00 004C 1022 .BYTE MFSK_FORMAT_NORMAL_11 ; NORMAL11.
05 004D 1023 .BYTE MFSK_FORMAT_IMAGE ; CORE-DUMP11 - IMAGE 11
01 004E 1024 .BYTE MFSK_FORMAT_NORMAL_15 ; NORMAL15
00 004F 1025 .BYTE MFSK_FORMAT_NORMAL_11 ; Illegal value defaults to NORMAL11.
```

```
.SBTTL TM78/TU78 FUNCTION DECISION TABLE
0050 1027      :+
0050 1028      : TM78/TU78 FUNCTION DECISION TABLE
0050 1029      :-
0050 1030      :
0050 1031      :
0050 1032      TF_FUNCNAME:
0050 1033      FUNCTAB
0050 1034      <NOP,-
0050 1035      UNLOAD,-
0050 1036      SPACERECORD,-
0050 1037      RECAL,-
0050 1038      DRVCLR,-
0050 1039      READPRESET,-
0050 1040      PACKACK,-
0050 1041      ERASETAPE,-
0050 1042      SENSECHAR,-
0050 1043      SETCHAR,-
0050 1044      SPACEFILE,-
0050 1045      WRITECHECK,-
0050 1046      WRITEPBLK,-
0050 1047      READPBLK,-
0050 1048      WRITEMARK,-
0050 1049      DSE,-
0050 1050      AVAILABLE,-
0050 1051      READLBLK,-
0050 1052      WRITELBLK,-
0050 1053      SENSEMODE,-
0050 1054      SETMODE,-
0050 1055      REWIND,-
0050 1056      REWINDOFF,-
0050 1057      SKIPRECORD,-
0050 1058      SKIPFILE,-
0050 1059      WRITEOF,-
0050 1060      READVBLK,-
0050 1061      WRITEVBLK,-
0050 1062      ACCESS,-
0050 1063      ACPCONTROL,-
0050 1064      CREATE,-
0050 1065      DEACCESS,-
0050 1066      DELETE,-
0050 1067      MODIFY,-
0050 1068      MOUNT>
0058 1069      FUNCTAB
0058 1070      <NOP,-
0058 1071      UNLOAD,-
0058 1072      SPACERECORD,-
0058 1073      AVAILABLE,-
0058 1074      RECAL,-
0058 1075      DRVCLR,-
0058 1076      READPRESET,-
0058 1077      PACKACK,-
0058 1078      ERASETAPE,-
0058 1079      SENSECHAR,-
0058 1080      SETCHAR,-
0058 1081      SPACEFILE,-
0058 1082      WRITEMARK,-
0058 1083      DSE,-

:FUNCTION DECISION TABLE
:LEGAL FUNCTIONS
:NO OPERATION
:UNLOAD VOLUME
:SPACE RECORDS
:RECALIBRATE (REWIND)
:DRIVE CLEAR
:READ IN PRESET
:PACK ACKNOWLEDGE
:ERASE TAPE
:SENSE TAPE CHARACTERISTICS
:SET CHARACTERISTICS
:SPACE FILE
:WRITE CHECK FORWARD
:WRITE PHYSICAL BLOCK
:READ PHYSICAL BLOCK
:WRITE TAPE MARK
:ERASE REST OF TAPE & REWIND **NEW FUNCTION
:AVAILABLE (REWIND/NOWAIT CLEAR VALID)
:READ LOGICAL BLOCK
:WRITE LOGICAL BLOCK
:SENSE TAPE MODE
:SET MODE
:REWIND
:REWIND AND SET OFFLINE
:SKIP RECORDS
:SKIP FILES
:WRITE END OF FILE
:READ VIRTUAL BLOCK
:WRITE VIRTUAL BLOCK
:ACCESS FILE AND/OR FIND DIRECTORY ENTRY
:ACP CONTROL FUNCTION
:CREATE FILE AND/OR CREATE DIRECTORY ENTRY
:DEACCESS FILE
:DELETE FILE AND/OR DIRECTORY ENTRY
:MODIFY FILE ATTRIBUTES
:MOUNT VOLUME
:BUFFERED I/O FUNCTIONS
:NO OPERATION
:UNLOAD VOLUME
:SPACE RECORDS
:AVAILABLE (REWIND/NOWAIT CLEAR VALID)
:RECALIBRATE (REWIND)
:DRIVE CLEAR
:READ IN PRESET
:PACK ACKNOWLEDGE
:ERASE TAPE
:SENSE CHARACTERISTICS
:SET CHARACTERISTICS
:SPACE FILES
:WRITE TAPE MARK
:ERASE REST OF TAPE & REWIND **NEW FUNCTION
```

0058	1084	SENSEMODE,-	:SENSE MODE
0058	1085	SETMODE,-	:SET MODE
0058	1086	REWIND,-	:REWIND
0058	1087	REWINDOFF,-	:REWIND AND UNLOAD
0058	1088	SKIPRECORD,-	:SKIP RECORDS
0058	1089	SKIPFILE,-	:SKIP FILES
0058	1090	WRITEOF,-	:WRITE END OF FILE
0058	1091	ACCESS,-	:ACCESS FILE AND/OR FIND DIRECTORY ENTRY
0058	1092	ACPCONTROL,-	:ACP CONTROL FUNCTION
0058	1093	CREATE,-	:CREATE FILE AND/OR CREATE DIRECTORY ENTRY
0058	1094	DEACCESS,-	:DEACCESS FILE
0058	1095	DELETE,-	:DELETE FILE AND/OR DIRECTORY ENTRY
0058	1096	MODIFY,-	:MODIFY FILE ATTRIBUTES
0058	1097	MOUNT>	:MOUNT VOLUME
0060	1098	FUNCTAB +ACPS\$READBLK,-	:READ FUNCTIONS
0060	1099	<READLBLK,-	:READ LOGICAL BLOCK FORWARD
0060	1100	READPBLK,-	:READ PHYSICAL BLOCK FORWARD
0060	1101	READVBLK>	:READ VIRTUAL BLOCK
006C	1102	FUNCTAB +ACPS\$WRITEBLK,-	:WRITE FUNCTIONS
006C	1103	<WRITECHECK,-	:WRITE CHECK FORWARD
006C	1104	WRITELBLK,-	:WRITE LOGICAL BLOCK
006C	1105	WRITEPBLK,-	:WRITE PHYSICAL BLOCK
006C	1106	WRITEVBLK>	:WRITE VIRTUAL BLOCK
0078	1107	FUNCTAB +ACPS\$ACCESS,<ACCESS,CREATE>	:ACCESS AND CREATE FILE OR DIRECTORY
0084	1108	FUNCTAB +ACPS\$DEACCESS,<DEACCESS>	:DEACCESS FILE
0090	1109	FUNCTAB +ACPS\$MODIFY,-	:
0090	1110	<ACPCONTROL,-	:ACP CONTROL FUNCTION
0090	1111	DELETE,-	:DELETE FILE OR DIRECTORY ENTRY
0090	1112	MODIFY>	:MODIFY FILE ATTRIBUTES
009C	1113	FUNCTAB +ACPS\$MOUNT,<MOUNT>	:MOUNT VOLUME
00A8	1114	FUNCTAB +MT\$CHECK ACCESS,-	:MAGTAPE CHECK ACCESS FUNCTIONS
00A8	1115	<ERASETAPE,-	:ERASE TAPE
00A8	1116	WRITEMARK,-	:WRITE TAPE MARK
00A8	1117	DSE,-	:ERASE REST OF TAPE & REWIND **NEW FUNCTION
00A8	1118	WRITEOF>	:WRITE END OF FILE
00B4	1119	FUNCTAB +EXE\$ZEROPARM,-	:ZERO PARAMETER FUNCTIONS
00B4	1120	<NOP,-	:NO OPERATION
00B4	1121	UNLOAD,-	:UNLOAD VOLUME
00B4	1122	RECAL,-	:RECALIBRATE (REWIND)
00B4	1123	REWIND,-	:REWIND
00B4	1124	REWINDOFF,-	:REWIND AND SET OFFLINE
00B4	1125	DRVCLR,-	:DRIVE CLEAR
00B4	1126	READPRESET,-	:READ IN PRESET
00B4	1127	PACKACK,-	:PACK ACKNOWLEDGE
00B4	1128	ERASETAPE,-	:ERASE TAPE
00B4	1129	SENSECHAR,-	:SENSE TAPE CHARACTERISTICS
00B4	1130	SENSEMODE,-	:SENSE TAPE MODE
00B4	1131	WRITEMARK,-	:WRITE TAPE MARK
00B4	1132	DSE,-	:ERASE REST OF TAPE & REWIND **NEW FUNCTION
00B4	1133	AVAILABLE,-	:AVAILABLE (REWIND/NOWAIT CLEAR VALID)
00B4	1134	WRITEOF>	:WRITE END OF FILE
00C0	1135	FUNCTAB +EXE\$ONEPARM,-	:ONE PARAMETER FUNCTIONS
00C0	1136	<SPACERECORD,-	:SPACE RECORDS
00C0	1137	SPACEFILE,-	:SPACE FILES
00C0	1138	SKIPRECORD,-	:SKIP RECORDS
00C0	1139	SKIPFILE>	:SKIP FILES
00CC	1140	FUNCTAB +EXE\$SETMODE,-	:SET TAPE CHARACTERISTICS

TFDRIVER
V04-000

- TM78/TU78 MAGTAPE DRIVER
TM78/TU78 FUNCTION DECISION TABLE

G 6

16-SEP-1984 00:08:15 VAX/VMS Macro V04-00
5-SEP-1984 00:17:37 [DRIVER.SRC]TFDRIVER.MAR;1

Page 25
(1)

TF
V0

00CC 1141
00CC 1142

<SETCHAR,-
SETMODE>

:
:

```
00D8 1144 .SBTTL CANCEL I/O ON CHANNEL
00D8 1145 :+
00D8 1146 : TF_CANCELIO - CANCEL I/O ON CHANNEL
00D8 1147 :
00D8 1148 : THIS ROUTINE IS CALLED WHEN THE LAST CHANNEL ASSIGNED TO A DEVICE IS DEASSIGNED,
00D8 1149 : THE DEVICE IS DEALLOCATED, AND WHEN THE CANCEL I/O ON CHANNEL SYSTEM SERVICE IS
00D8 1150 : EXECUTED.
00D8 1151 :
00D8 1152 : INPUTS:
00D8 1153 :
00D8 1154 : R2 = NEGATIVE CHANNEL NUMBER.
00D8 1155 : R3 = ADDRESS OF CURRENT I/O REQUEST PACKET.
00D8 1156 : R4 = CURRENT PROCESS PCB ADDRESS.
00D8 1157 : R5 = DEVICE UCB ADDRESS.
00D8 1158 :
00D8 1159 : OUTPUTS:
00D8 1160 :
00D8 1161 : IF THE DEVICE IS CURRENTLY BUSY, DOING A REWIND, AND IN A WAITFOR INTERRUPT
00D8 1162 : STATE, THEN THE REWIND FUNCTION IS CANCELLED.
00D8 1163 : -
00D8 1164 :
00D8 1165 TF_CANCELIO:
00D8 1166 JSB G^IOC$CANCELIO ;CANCEL I/O ON CHANNEL
00DE 1167 BBC #UCBSV CANCEL,- ;TEST IF FUNCTION SHOULD BE CANCELLED
00E0 1168 UCBSW_STS(R5),30$ ;IF CLR, NO CANCEL PENDING
00E3 1169 DSBINT ;DISABLE INTERRUPTS
00E9 1170 BBC #UCBSV MF POWER,- ; See if currently repositioning tape
00EB 1171 UCBSW_DEVSTS(R5),5$ ; due to POWERFAIL. If NOT, branch around
00EE 1172 BISW #UCBSM MF CNCLP,- ; If SO, set CANCEL PENDING flag on.
00F0 1173 UCBSW_DEVSTS(R5) ; This bit is needed only if the driver
00F2 1174 ; thread in POWERFAIL is currently
00F2 1175 ; not waiting for an interrupt. That
00F2 1176 ; would imply that the driver thread
00F2 1177 ; is on a resource wait queue and
00F2 1178 ; would have to abort the operation
00F2 1179 ; itself.
00F2 1180
00F2 1181 BITW #UCBSM_INT!UCBSM_TIM,-
00F4 1182 UCBSW_STS(R5) ; Interrupt or timeout expected?
00F6 1183 BEQL 20$ ; If NOT, branch around and let CANCEL
00F8 1184 ; PENDING flag advise repositioning
00F8 1185 ; thread of the need to ABORT operation.
00F8 1186
00F8 1187 BICW #UCBSM MF POWER!UCBSM_MF_CNCLP,-
00FA 1188 UCBSW_DEVSTS(R5) ; Here we are awaiting interrupt; so
00FC 1189 ; we can safely kill the driver thread.
00FC 1190
00FC 1191 BRB 15$ ; Branch around.
00FE 1192 5$:
00FE 1193 CMPB #F REWIND!GO BIT,- ; Was last command executed a REWIND
0100 1194 UCBSL_MF_CMD(R5)
0103 1195 BEQL 10$ ;IF EQL YES
0105 1196 CMPB #F DSE!GO BIT,- ; Or was it a DSE (ERASE REST OF TAPE
0107 1197 UCBSL_MF_CMD(R5) ; AND REWIND) command?
010A 1198 BEQL 10$ ; If EQL yes.
010C 1199 BBC #UCBSV MF REWIND,- ; Is a REWIND in progress holding up
010E 1200 UCBSW_DEVSTS(R5),20$ ; current IRP? If NOT, branch.
```

00000000'GF 16 00D8 1166 JSB G^IOC\$CANCELIO ;CANCEL I/O ON CHANNEL
03 E1 00DE 1167 BBC #UCBSV CANCEL,- ;TEST IF FUNCTION SHOULD BE CANCELLED
4E 64 A5 00E0 1168 UCBSW_STS(R5),30\$;IF CLR, NO CANCEL PENDING
01 E1 00E3 1169 DSBINT ;DISABLE INTERRUPTS
10 68 A5 00E9 1170 BBC #UCBSV MF POWER,- ; See if currently repositioning tape
04 A8 00EB 1171 UCBSW_DEVSTS(R5),5\$; due to POWERFAIL. If NOT, branch around
68 A5 00EE 1172 BISW #UCBSM MF CNCLP,- ; If SO, set CANCEL PENDING flag on.
00F0 1173 UCBSW_DEVSTS(R5) ; This bit is needed only if the driver
00F2 1174 ; thread in POWERFAIL is currently
00F2 1175 ; not waiting for an interrupt. That
00F2 1176 ; would imply that the driver thread
00F2 1177 ; is on a resource wait queue and
00F2 1178 ; would have to abort the operation
00F2 1179 ; itself.
03 B3 00F2 1180
64 A5 B3 00F2 1181 BITW #UCBSM_INT!UCBSM_TIM,-
36 13 00F4 1182 UCBSW_STS(R5) ; Interrupt or timeout expected?
00F6 1183 BEQL 20\$; If NOT, branch around and let CANCEL
00F8 1184 ; PENDING flag advise repositioning
00F8 1185 ; thread of the need to ABORT operation.
00F8 1186
06 AA 00F8 1187 BICW #UCBSM MF POWER!UCBSM_MF_CNCLP,-
68 A5 00FA 1188 UCBSW_DEVSTS(R5) ; Here we are awaiting interrupt; so
00FC 1189 ; we can safely kill the driver thread.
00FC 1190
19 11 00FC 1191 BRB 15\$; Branch around.
07 91 00FE 1192 5\$:
0180 C5 0100 1193 CMPB #F REWIND!GO BIT,- ; Was last command executed a REWIND
0C 13 0103 1194 UCBSL_MF_CMD(R5)
08 91 0105 1195 BEQL 10\$;IF EQL YES
0180 C5 0107 1196 CMPB #F DSE!GO BIT,- ; Or was it a DSE (ERASE REST OF TAPE
05 13 010A 1197 UCBSL_MF_CMD(R5) ; AND REWIND) command?
00 E1 010C 1198 BEQL 10\$; If EQL yes.
1D 68 A5 010E 1199 BBC #UCBSV MF REWIND,- ; Is a REWIND in progress holding up
010E 1200 UCBSW_DEVSTS(R5),20\$; current IRP? If NOT, branch.

03	B3	0111	1201	10\$:	BITW	#UCBSM_INT:UCBSM_TIM,-	
64	A5	0113	1202			UCBSW_STS(R5)	; INTERRUPT OR TIMEOUT EXPECTED?
17	13	0115	1203		BEQL	20\$	; IF EQL NO
		0117	1204	15\$:			
03	AA	0117	1205		BICW	#UCBSM_INT:UCBSM_TIM,-	
64	A5	0119	1206			UCBSW_STS(R5)	; CLEAR INTERRUPT AND TIMEOUT EXPECTED
01	A8	011B	1207		BISW	#UCBSM_MF_REWIND,-	
68	A5	011D	1208			UCBSW_DEVSTS(R5)	; SET REWIND IN PROGRESS
		011F	1209		SETIPL	UCBSB_FIPL(R5)	; LOWER TO FORK LEVEL
50	2C	3C	0123	1210	MOVZWL	#SSS_ABORT,R0	; SET ABORT STATUS
	54	DD	0126	1211	PUSHL	R4	; SAVE CURRENT PROCESS PCB ADDRESS
12AB	30	0128	1212		BSBW	STXIT	; FINISH I/O OPERATION
54	8ED0	012B	1213		POPL	R4	; RESTORE CURRENT PROCESS PCB ADDRESS
		012E	1214	20\$:	ENBINT		; LOWER TO FORK LEVEL
	05	0131	1215	30\$:	RSB		;

```
0132 1217 .SBTTL START I/O OPERATION (and RESTARTIO)
0132 1218 :+
0132 1219 : TF_STARTIO - START I/O OPERATION ON DEVICE UNIT
0132 1220 :
0132 1221 : THIS ENTRY POINT IS ENTERED TO START AN I/O OPERATION ON A DEVICE UNIT.
0132 1222 :
0132 1223 : INPUTS:
0132 1224 :
0132 1225 : R3 = ADDRESS OF I/O PACKET.
0132 1226 : R5 = UCB ADDRESS OF DEVICE UNIT.
0132 1227 :
0132 1228 : OUTPUTS:
0132 1229 :
0132 1230 : FUNCTION DEPENDENT PARAMETERS ARE STORED IN THE DEVICE UCB, THE ERROR
0132 1231 : RETRY COUNT IS RESET, AND THE FUNCTION IS EXECUTED. AT FUNCTION COMPLETION
0132 1232 : THE OPERATION IS TERMINATED THROUGH REQUEST COMPLETE.
0132 1233 :-
0132 1234 :
0132 1235 : .ENABLE LSB
0132 1236 TF_RESTARTIO:
0132 1237 : Come here to setup registers to
: RESTART the current I/O operation.
: Remember # times restarted.
: R3 => IRP.
: Initialize UCB fields
00BB C5 96 0132 1238 INCB UCBSB_MF_RSTCNT(R5)
53 58 A5 D0 0136 1239 MOVL UCBSL_IRP(R5),R3
2C A3 7D 013A 1240 MOVQ IRPSL_SVAPTE(R3),-
78 A5 013D 1241 UCBSL_SVAPTE(R5)
05 91 013F 1242 CMPB #MAX_RESTARTIO,-
00BB C5 0141 1243 UCBSB_MF_RSTCNT(R5)
0C 18 0144 1244 BGEQ 10$
50 0054 8F 3C 0146 1245 MOVZWL #SS$ CTRLERR,R0
1244 31 014B 1246 BRW FUNCTION_EXIT
014E 1247 : Prevent endless loop at fork level
: from hanging system.
: GEQ implies that we may try again.
: Else indicate controller problem.
: And terminate.
00BB C5 94 014E 1248 TF_STARTIO:
014E 1249 CLRB UCBSB_MF_RSTCNT(R5)
: START I/O OPERATION
: Clear RESTARTIO count.
0040 8F AA 0152 1250 10$: BICW #UCBSM_MF_UCBFREE,-
68 A5 0156 1252 UCBSW_DEVSTS(R5)
07 E1 0158 1253 BBC #UCBSV_MF_UCBBUSY,-
01 68 A5 015A 1254 UCBSW_DEVSTS(R5),20$
05 015D 1255 RSB
: Indicate that UCB FORK block cannot
: be used asynchronously for now.
: If UCB FORK block NOT currently in use
: then branch around to continue.
: If FORK block in use, return to caller
00E4 C5 9E 015E 1256 20$: MOVAB UCBSL_MF_SUBSTACK(R5),-
00E0 C5 0162 1258 UCBSL_MF_SUBSP(R5)
05 E1 0165 1259 BBC #UCBSV_MF_ATTN,-
09 68 A5 0167 1260 UCBSW_DEVSTS(R5),30$
1704 30 016A 1261 BSBW ISSUE_TMCLR
03 50 E8 016D 1262 BLBS R0,30$
121F 31 0170 1263 BRW FUNCTION_EXIT
: Initialize SUBSTACK pointer to base
: of SUBSTACK array.
: Bit set implies we must CLEAR the
: TM78. Clear implies continue.
: Call to clear TM78.
: LBS implies successful TM78 clearing.
: If we couldn't reset TM78, exit.
04 E1 0173 1265 30$: BBC #UCBSV_MF_TMRDY,-
03 68 A5 0175 1266 UCBSW_DEVSTS(R5),40$
16B0 30 0178 1267 BSBW AWAIT_TMRDY
: Bit set implies that we must await
: TM78 coming ready, Clear implies AOK.
: Wait for TM78 to come READY.
20 A3 B0 017B 1268 40$: MOVW IRPSW_FUNC(R3),-
009A C5 017E 1270 UCBSW_FUNC(R5)
0081 C5 90 0181 1271 MOVB UCBSB_ERTMAX(R5),-
0080 C5 0185 1272 UCBSB_ERTCNT(R5)
28 BB 0188 1273 PUSHF #M<R3,R5>
: Save function code and modifiers.
: Copy maximum retry count to error
: retry count.
: Save registers destroyed by MOVC5.
```

```
00A4 8F 00 FE AF 00 2C 018A 1274      MOVCS  #0,,#0,UCB REGDUMP_LEN,-
          0120 C5      0192 1275      UCB$M_MFMBA_CSR(R5) : Clear out Register Dump area in UCB.
          28 BA 0195 1276      POPR  #^M<R3,R5> : Restore input registers.
          0200 8F AA 0197 1277
          68 A5 AA 0197 1278      BICW  #UCB$M_MF_EXSNS_DONE,- : Clear bit so as to allow EXTENDED
          AA 0198 1279      UCB$W_DEVSTS(R5) : SENSE to proceed.
          019D 1280      BICW  #<MTSM_BOT!- : Clear beginning of tape and,
          019E 1281      MTSM_EOF!- : End of File and
          019E 1282      MTSM_EOT>-16,- : End of Tape
          46 A5 07 019E 1283      UCB$M_DEVDEPEND+2(R5)
          50 01 CE 01A1 1284      MNEGL #1,R0 : Prepare to initialize UCB$M_MF_ORGPOS
          51 02 CE 01A4 1285      MNEGL #2,R1 : and UCB$M_MF_ORGPTM with the values
          08 E1 01A7 1286      BBC  #UCB$V_VALID,- : in UCB$M_RECORD and UCB$M_MF_PREVTM.
          0A 64 A5 01A9 1287      UCB$W_STS(R5),50$ : However if invalid, leave them set to
          01AC 1288 : -1 and -2 respectively.
          50 00B0 C5 D0 01AC 1289      MOVL UCB$M_RECORD(R5),R0 : Pickup original position in volume.
          51 00C8 C5 D0 01B1 1290      MOVL UCB$M_MF_PREVTM(R5),R1 : Also position of last TAPEMARK.
          00C4 C5 50 D0 01B6 1291 50$:      MOVL R0,UCB$M_MF_ORGPOS(R5) : Remember original position and
          00CC C5 51 D0 01BB 1292      MOVL R1,UCB$M_MF_ORGPTM(R5) : original previous TAPEMARK.
          08 E0 01C0 1293      BBS  #IRP$V_PHYSIO,- :
          0D 2A A3 01C2 1295      BBS  #UCB$V_VALID,- : IF SET, PHYSICAL I/O FUNCTION
          08 64 A5 01C7 1297      UCB$W_STS(R5),60$ : IF SET, VOLUME SOFTWARE VALID
          50 0254 8F 3C 01CA 1298      MOVZWL #SS$VOLINV,R0 : SET VOLUME INVALID STATUS
          11C0 31 01CF 1299      BRW  FUNCTION_EXIT :
          00 EF 01D2 1300 60$:      EXTZV #IRP$V_FCODE,- : Extract I/O function code and
          06 01D4 1302      #IRP$S_FCODE,- : place code into a register that
          54 20 A3 01D5 1303      IRP$W_FUNC(R3),R4 : will survive an IOFORK.
          01D8 1304
          01D8 1305
          53 24 A5 D0 01D8 1306      MOVL UCB$M_CRB(R5),R3 : R3=>TM78-CRB.
          01DC 1307      ASSUME IDB$M_CSR EQ 0
          53 2C B3 D0 01DC 1308      MOVL @CRB$C_INTD+VEC$M_IDB(R3),R3 : R3=>TM78-CSR.
          01E0 1309
          01E0 1310 :
          01E0 1311 : Here we wait if an unfinished REWIND is in progress.
          01E0 1312 :
          01E0 1313 :
          01E0 1314 70$:
          01E0 1315      DSBINT : Block interrupts
          01E6 1316      PRIOR_TEST_TMRDY : Assure that TM78 ready.
          01EF 1317
          27 64 A5 E0 01EF 1318      BBS  #UCB$V_POWER,- : Branch if there has been a POWERFAIL
          00 E1 01F1 1319      UCB$W_STS(R5),80$ : Branch around WAIT if NO REWIND in
          2F 68 A5 01F4 1320      BBC  #UCB$V_MF_REWIND,- : progress.
          01F6 1321      UCB$W_DEVSTS(R5),100$ : Wait for REWIND to finish.
          01F9 1322      WFIKPC 90$,#TU78_MAX_DSE : Label of return point for this WFIKPC.
          0207 1323      WAITING_FOR_REWIND: : Assure that TM78 ready.
          0207 1324      POST_TEST_TMRDY : Update UCB after REWIND.
          1422 30 0210 1325      BSBW TF_UNSOLONT
          C5 11 0213 1326      IOFORK
          0219 1327      BRB 70$ : Branch back to test REWIND bit again.
          0218 1328
          0218 1329 80$:      ENBINT
          021E 1330 90$:      SETIPL UCB$B_FIPL(R5) : Set to fork IPL.
```



```
0C51 30 0222 1331      BSBW  TIMEOUT POWERFAIL      ; Call to indicate problem.
116A 31 0225 1332      BRW   FUNCTION_EXIT           ; If we experience return, goto EXIT.
                                0228 1333 100$:
                                0228 1334      ENBINT           ; Enable interrupts if REWIND done.
51 54 00 022B 1335      MOVL   R4,R1                  ; Copy I/O function code to R1.
                                022E 1336
                                022E 1337      ; Now we have the I/O function code without modifiers in R1. We use it to
                                022E 1338      ; branch to the particular code that handles the function.
                                022E 1339
                                022E 1340
                                022E 1341      ASSUME  IOS_NOP           EQ  0
                                022E 1342      ASSUME  IOS_UNLOAD        EQ  1
                                022E 1343      ASSUME  IOS_SPACEFILE     EQ  2
                                022E 1344      ASSUME  IOS_RECAL         EQ  3
                                022E 1345      ASSUME  IOS_DRVCLR        EQ  4
                                022E 1346      ASSUME  IOS_ERASETAPE     EQ  6
                                022E 1347      ASSUME  IOS_PACKACK       EQ  8
                                022E 1348      ASSUME  IOS_SPACERECORD    EQ  9
                                022E 1349      ASSUME  IOS_WRITECHECK    EQ 10
                                022E 1350      ASSUME  IOS_WRITEPBLK     EQ 11
                                022E 1351      ASSUME  IOS_READPBLK      EQ 12
                                022E 1352      ASSUME  IOS_DSE           EQ 21
                                022E 1353      ASSUME  IOS_AVAILABLE    EQ 17
                                022E 1354      ASSUME  IOS_READPRESET    EQ 25
                                022E 1355      ASSUME  IOS_SETCCHAR      EQ 26
                                022E 1356      ASSUME  IOS_SENSECHAR     EQ 27
                                022E 1357      ASSUME  IOS_WRITEMARK     EQ 28
                                022E 1358      ASSUME  IOS_WRITEBLK      EQ 32
                                022E 1359      ASSUME  IOS_READLBLK      EQ 33
                                022E 1360      ASSUME  IOS_REWINDOFF     EQ 34
                                022E 1361      ASSUME  IOS_SETMODE       EQ 35
                                022E 1362      ASSUME  IOS_REWIND        EQ 36
                                022E 1363      ASSUME  IOS_SKIPFILE      EQ 37
                                022E 1364      ASSUME  IOS_SKIPRECORD    EQ 38
                                022E 1365      ASSUME  IOS_SENSEMODE     EQ 39
                                022E 1366      ASSUME  IOS_WRITEOF       EQ 40
                                022E 1367      CASE      R1,2-
                                022E 1368      START_NOP,-
                                022E 1369      START_UNLOAD,-
                                022E 1370      START_SPACEFILE,-
                                022E 1371      START_RECAL,-
                                022E 1372      START_DRVCLR,-
                                022E 1373      START_NOSUCH,-
                                022E 1374      START_ERASETAPE,-
                                022E 1375      START_NOSUCH,-
                                022E 1376      START_PACKACK,-
                                022E 1377      START_SPACERECORD,-
                                022E 1378      START_WRITECHECK,-
                                022E 1379      START_WRITEPBLK,-
                                022E 1380      START_READPBLK,-
                                022E 1381      START_NOSUCH,-
                                022E 1382      START_NOSUCH,-
                                022E 1383      START_NOSUCH,-
                                022E 1384      START_NOSUCH,-
                                022E 1385      START_AVAILABLE,-
                                022E 1386      START_NOSUCH,-
                                022E 1387      START_NOSUCH,-
```

```
: 0
: 1
: 2
: 3
: 4
: 5
: 6
: 7
: 8
: 9
: 10
: 11
: 12
: 13
: 14
: 15
: 16
: 17
: 18
: 19
```

```
022E 1388          START_NOSUCH,-          : 20
022E 1389          START_DSE,-             : 21
022E 1390          START_NOSUCH,-          : 22
022E 1391          START_NOSUCH,-          : 23
022E 1392          START_NOSUCH,-          : 24
022E 1393          START_READPRESET,-      : 25
022E 1394          START_SETCCHAR,-        : 26
022E 1395          START_SENSECHAR,-       : 27
022E 1396          START_WRITEMARK,-       : 28
022E 1397          START_NOSUCH,-          : 29
022E 1398          START_NOSUCH,-          : 30
022E 1399          START_NOSUCH,-          : 31
022E 1400          START_WRITEBLK,-        : 32
022E 1401          START_READBLK,-         : 33
022E 1402          START_REWINDOFF,-       : 34
022E 1403          START_SETMODE,-         : 35
022E 1404          START_REWIND,-          : 36
022E 1405          START_SKIPFILE,-        : 37
022E 1406          START_SKIPRECORD,-      : 38
022E 1407          START_SENSEMODE,-       : 39
022E 1408          START_WRITEOF,-        : 40
022E 1409          >
0284 1410          START_NOSUCH:
50 00F4 8F 3C 0284 1411          MOVZWL #SS$ ILLIOFUNC,RO
   114A 31 0289 1412          BRW STSXT
028C 1413          .DISABLE LSB          ; Branch to exit I/O function.
```

```
028C 1415 .SBTTL Start NOP, DRVCLR, PACKACK, SENSECHAR or SENSEMODE I/O functions
028C 1416
028C 1417 ; START_NOP, START_DRVCLR, PACKACK, SENSECHAR and SENSEMODE
028C 1418
028C 1419 ; INPUTS:
028C 1420 ; R3 => TM78 CSR
028C 1421 ; R5 => UCB
028C 1422 ;
028C 1423
028C 1424 START_SENSEMODE:
028C 1425 START_SENSECHAR:
028C 1426 START_DRVCLR:
028C 1427 START_PACKACK:
028C 1428 START_NOP:
028C 1429 EX_NDT_CMD CMD=SENSE,- ; Execute a SENSE command
028C 1430 TIMEOUT=#MINIMUM_TIMEOUT,-; which should not take much time
028C 1431 ERROR_LABEL=NOP_EXIT
029B 1432
029B 1433 ;
029B 1434 ; EX_NDT_CMD returns the interrupt code in R0. The only possible interrupt
029B 1435 ; codes that can occur after a SENSE command are:
029B 1436 ;
029B 1437 ; 1. HIC_DONE - which implies a successful completion.
029B 1438 ; 2. HIC_TM_FAULT_A - which implies a controller error.
029B 1439 ; 3. HIC_TU_FAULT_A - which implies a drive error.
029B 1440 ;
029B 1441
50 01 91 029B 1442 CMPB #HIC_DONE,R0 ; Was it successful?
24 12 029E 1443 BNEQ 20$ ; If not, branch to see why not.
1444 30 02A0 1444 BSBW RECORD_SENSE_INFO ; Update UCB fields with latest data.
0F E0 02A3 1445 BBS #MF_DS-V_RDY,- ; If tape physically mounted,
07 0158 C5 02A5 1446 UCB$#MF_DS(R5),5$ ; branch around.
50 01A4 8F 3C 02A9 1447 MOVZWL #SS$#MEDOFL,R0 ; Set proper return status.
28 11 02AE 1448 BRB NOP_EXIT ; Branch around to exit.
00 ED 02B0 1449 5$:
06 02B0 1450 CMPZV #IRPSV_FCODE,- ; See if the active I/O operation is
009A C5 02B2 1451 #IRPSS_FCODE,- ; a PACKACK function and if NOT then
08 02B3 1452 UCB$#FUNC(R5),- ; we will branch around.
06 12 02B6 1453 #IOS_PACKACK
0800 8F A8 02B7 1454 BNEQ 10$ ; Branch if not PACKACK.
64 A5 02B9 1455 BISW #UCB$#VALID,- ; For successful PACKACK functions we
02BD 1456 UCB$#STS(R5) ; set the volume software valid.
50 01 3C 02BF 1457 10$:
14 11 02BF 1458 MOVZWL S^#SS$#NORMAL,R0 ; Else indicate success status.
0F06 30 02C2 1459 BRB NOP_EXIT ; And branch around to exit.
50 18 91 02C4 1460 20$:
07 12 02C4 1461 BSBW DEVICE_ERROR ; Do EXTENDED SENSE and log device error
50 0054 8F 3C 02C7 1462 CMPB #HIC_TM_FAULT_A,R0 ; Was it a controller failure?
05 11 02CA 1463 BNEQ 30$ ; If not, go to indicate DRIVE error.
50 008C 8F 3C 02CC 1464 MOVZWL #SS$#CTRLERR,R0 ; Else indicate controller failure.
10B7 31 02D1 1465 BRB NOP_EXIT ; And branch around to exit.
02D3 1466
50 008C 8F 3C 02D3 1467 30$: MOVZWL #SS$#DRVERR,R0 ; Indicate drive error.
02D8 1468 NOP_EXIT:
02D8 1469 BRW FUNCTION_EXIT ; Branch to common exit.
```

```
02DB 1471 .SBTTL Start REWIND, RECAL, READPRESET, REWINDOFF or UNLOAD function.
02DB 1472
02DB 1473 : START_REWIND, START_RECAL, START_READPRESET, START_REWINDOFF and START_UNLOAD
02DB 1474 : and START_AVAILABLE
02DB 1475
02DB 1476 : INPUTS:
02DB 1477 : R3 => TM78 CSR
02DB 1478 : R5 => UCB
02DB 1479 :
02DB 1480
02DB 1481 START_AVAILABLE:
0080 8F A8 02DB 1482 BISW #IOSM_NOWAIT,- ; Available is equivalent of REWIND-
009A C5 02DB 1483 UCBSW_FUNC(R5) ; NOWAIT and clear UCBSM_VALID.
0800 8F AA 02E2 1484 BICW #UCBSM_VALID,- ; Mark volume as invalid
64 A5 02E6 1485 UCBSW_STS(R5) ; and fall thru to REWIND.
02E8 1486
02E8 1487 START_REWIND:
02E8 1488 START_RECAL:
02E8 1489 START_READPRESET:
50 07 9A 02E8 1490 MOVZBL #F_REWIND!GO_BIT,R0 ; Set command code in register.
03 11 02EB 1491 BRB REWIND_COMMON ; Branch around to common code.
02ED 1492
02ED 1493 START_REWINDOFF:
02ED 1494 START_UNLOAD:
50 05 9A 02ED 1495 MOVZBL #F_UNLOAD!GO_BIT,R0 ; Set command code in register.
02F0 1496 REWIND_COMMON:
0092 C5 50 90 02F0 1497 MOVB R0,UCBSB_FEX(R5) ; Save command in convenient place.
02F5 1498 REWIND CMD=R0,- ; Execute REWIND or UNLOAD.
02F5 1499 WAIT=UCBSW_FUNC(R5),- ; Pass value of IOSM_NOWAIT.
02F5 1500 ERROR_LABEL=REWIND_EXIT
02FF 1501
02FF 1502 :
02FF 1503 : REWIND returns the interrupt code in R0.
02FF 1504 :
02FF 1505 : The only possible interrupt codes here are:
02FF 1506 :
02FF 1507 : 1. HIC_DONE
02FF 1508 : 2. HIC_NOT_AVAIL
02FF 1509 : 3. HIC_OFF_LINE
02FF 1510 : 4. HIC_NON_EX
02FF 1511 : 5. HIC_TM_FAULT_A
02FF 1512 : 6. HIC_TU_FAULT_A
02FF 1513 :
02FF 1514 :
50 01 91 02FF 1515 CMPB #HIC_DONE,R0 ; See if DONE,
23 12 0302 1516 BNEQ 10$ ; If NOT, branch to test interrupt code.
05 91 0304 1517 CMPB #F_UNLOAD!GO_BIT,- ; See if tape has been unloaded and
0092 C5 0306 1518 UCBSB_FEX(R5) ; if so then we mark the volume as
0309 1519 ; invalid.
06 12 0309 1520 BNEQ 5$ ; If NOT UNLOAD, branch around.
0800 8F AA 0308 1521 BICW #UCBSM_VALID,- ; Mark volume as invalid.
64 A5 030F 1522 UCBSW_STS(R5)
0311 1523 5$:
50 01 3C 0311 1524 MOVZWL S^#SS$-NORMAL,R0 ; Indicate success code and
01 A8 0314 1525 BISW #<MTSM_BOT@-16>,- ; Set BOT bit on in UCB field.
46 A5 0316 1526 UCBSL_DEVDEPEND+2(R5)
10 AA 0318 1527 BICW #<MTSM_LOST@-16>,- ; Clear the fact that we may have
```

```

      46 A5      031A 1528      UCB$$_DEVDEPEND+2(R5) ; been lost
00C8 00B0 C5    D4 031C 1529      CLRL UCB$$_RECORD(R5) ; Indicate tape is positioned at start.
      C5      02    CE 0320 1530      MNEGL #2,UCB$$_MF_PREVTM(R5) ; And indicate ignorance of TAPEMARK positio
      2C      11    0325 1531      BRB REWIND_EXIT ; branch around to exit.
      0327 1532
      0327 1533 10$:
      50 0B      91 0327 1534      CMPB #HIC_OFF_LINE,R0 ; See if OFF-LINE.
      07      12 032A 1535      BNEQ 20$ ; If not, branch around to test further.
50 71A4 8F      3C 032C 1536      MOVZWL #$$$_MEDOFL,R0 ; Indicate media off-line and
      20      11 0331 1537      BRB REWIND_EXIT ; branch around to end.
      0333 1538
      0333 1539 20$:
      0E97      30 0333 1540      BSBW DEVICE_ERROR ; Do EXTENDED SENSE and log device error
      50 18      91 0336 1541      CMPB #HIC_TM_FAULT_A,R0 ; See if controller error.
      07      12 0339 1542      BNEQ 30$ ; If not, branch around to next test.
50 0054 8F      3C 033B 1543      MOVZWL #$$$_CTRLERR,R0 ; Indicate controller error and
      11      11 0340 1544      BRB REWIND_EXIT ; branch around.
      0342 1545
      50 0C      91 0342 1546 30$:      CMPB #HIC_NON_EX,R0 ; Test if non-existent drive.
      07      12 0345 1547      BNEQ 40$ ; If not then branch to drive error.
50 01C4 8F      3C 0347 1548      MOVZWL #$$$_NONEXDRV,R0 ; Indicate non-existent drive and
      05      11 034C 1549      BRB REWIND_EXIT ; branch around to exit.
      034E 1550
50 008C 8F      3C 034E 1551 40$:      MOVZWL #$$$_DRVERR,R0 ; Indicate drive error
      0353 1552
      0353 1553 REWIND_EXIT:
      103C      31 0353 1554      BRW FUNCTION_EXIT ; Branch to common exit.
```

```
0356 1556 .SBTTL Start a SETCHAR or a SETMODE function
0356 1557
0356 1558 : START_SETCHAR and START_SETMODE
0356 1559 : The quad-word of data for the operation is contained in IRPSL_MEDIA.
0356 1560 : This 'PHYSICAL' I/O function and the 'LOGICAL' I/O function
0356 1561 : SET MODE are almost identical. The only difference is that while
0356 1562 : both allow for the setting of:
0356 1563 :
0356 1564 :     1. Default buffer size
0356 1565 :     2. Tape density (1600 BPI or 6250 BPI).
0356 1566 :     3. Tape format
0356 1567 :
0356 1568 : the former function (i.e. SET CHARACTERISTICS) also allows for
0356 1569 : the resetting of the DEVICE CLASS and the DEVICE TYPE fields in
0356 1570 : the UCB.
0356 1571 :
0356 1572 : The first two bytes of the QUADWORD of data at IRPSL_MEDIA contain
0356 1573 : the DEVICE CLASS and DEVICE TYPE respectively for a SETCHAR.
0356 1574 : The next word of the QUADWORD contains the new buffer size. The
0356 1575 : third word contains new density and format information. The fourth
0356 1576 : word of the QUADWORD is reserved.
0356 1577 :
0356 1578 : INPUTS:
0356 1579 :     R3 => TM78 CSR
0356 1580 :     R5 => UCB
0356 1581 :
0356 1582 :
0356 1583 START_SETCHAR:
52 58 A5 D0 0356 1584     MOVL    UCBSL_IRP(R5),R2                ; R2 => current IRP.
40 A5 38 A2 B0 035A 1585     ASSUME   UCBSB_DEVTYPE EQ UCBSB_DEVCLASS+1
035A 1586     MOVW    IRPSL_MEDIA(R2),UCBSB_DEVCLASS(R5)    ; Reset CLASS and TYPE.
035F 1587
035F 1588 START_SETMODE:
52 58 A5 D0 035F 1589     MOVL    UCBSL_IRP(R5),R2                ; R2 => current IRP.
42 A5 3A A2 B0 0363 1590     MOVW    IRPSL_MEDIA+2(R2),UCBSW_DEVBUSIZ(R5)    ; Copy new buffer size.
7C A5 3C A2 B0 0368 1591     MOVW    IRPSL_MEDIA+4(R2),UCBSW_BOFF(R5)        ; Copy density, format
036D 1592                                     ; to unused UCB field
036D 1593                                     ; for now.
036D 1594
036D 1595     EX_NDT_CMD    CMD=SENSE,-          ; Execute Non-data transfer SENSE command
036D 1596                     TIMEOUT=#MINIMUM TIMEOUT,-; don't wait for long, and
036D 1597                     ERROR_LABEL=SETMODE_EXIT
037C 1598                                     ; Branch around to EXIT if unable.
037C 1599
037C 1600 : Remember EX_NDT_CMD returns the interrupt code in R0.
037C 1601
50 01 91 037C 1602     CMPB    #HIC_DONE,R0                ; Compare to success code.
16 13 037F 1603     BEQL    20$                    ; If no error, branch.
0381 1604
0E49 30 0381 1605     BSBW    DEVICE_ERROR                ; Do EXTENDED SENSE and log device error
0384 1606
50 19 91 0384 1607     CMPB    #HIC_TU_FAULT_A,R0            ; See if DRIVE error.
07 13 0387 1608     BEQL    10$                    ; Branch if yes.
50 0054 8F 3C 0389 1609     MOVZWL  #SS$_CTRLERR,R0            ; Indicate CONTROLLER error.
69 11 038E 1610     BRB      SETMODE_EXIT                ; Branch around to exit.
0390 1611
0390 1612 10$:
```

```
50 008C 8F 3C 0390 1613 MOVZWL #SS$_DRVERR,R0 ; Indicate DRIVE error.
    62 11 0395 1614 BRB SETMODE_EXIT ; And branch around to exit.
    0397 1615
    0397 1616 20$:
    0397 1617
    0397 1618
    0B 7C 0F E1 0397 1619 BBC #MT$V_LOGSOFTOG,- ; Test if caller has specified the
    06 44 0E E3 0399 1620 UCBSW_BOFF(R5),25$ ; toggling of MT$M_LOGSOFT in
    4000 8F AA 039C 1621 ; UCBSL_DEVDEPEND. If NOT, branch.
    44 A5 039E 1622 BBBS #MT$V_LOGSOFT,- ; Toggle bit. Here if clear, set it
    03A1 1623 UCBSL_DEVDEPEND(R5),25$ ; and branch around.
    03A5 1624 BICW #MT$M_LOGSOFT,- ; Here it was set so we clear it.
    03A7 1625 25$:
    133D 30 03A7 1627 BSBW RECORD_SENSE_INFO ; Call to record the sense information.
    08 EF 03AA 1628 EXTZV #MT$V_DENSITY,-
    05 03AC 1629 #MT$S_DENSITY,-
    50 7C A5 03AD 1630 UCBSW_BOFF(R5),R0 ; Extract user designated DENSITY parameter.
    04 EF 03B0 1631 EXTZV #MT$V_FORMAT,-
    04 03B2 1632 #MT$S_FORMAT,-
    51 7C A5 03B3 1633 UCBSW_BOFF(R5),R1 ; Likewise the FORMAT parameter
    03B6 1634
    51 FC85 CF41 9A 03B6 1636 MOVZBL FORMAT_TABLE[R1],R1 ; Replace software FORMAT with TM78
    03BC 1637 ; specific code.
    03BC 1638
    52 00F0 8F 3C 03BC 1639 MOVZWL #MT$M_FORMAT,R2 ; Set up mask to reset UCB field
    0A E1 03C1 1640 BBC #MF_DS_V_BOT,-
    18 0158 C5 03C3 1641 UCBSL_MF_DS(R5),30$ ; We only reset the DENSITY if we are at
    03C7 1642 ; beginning of tape. Else branch around.
    03C7 1643
    52 1F00 8F A8 03C7 1645 BISW #MT$M_DENSITY,R2 ; Set mask to accept DENSITY.
    00BA C5 94 03CC 1646 CLRB UCBSB_MF_DENSITY(R5) ; Initialize UCB DENSITY byte.
    50 04 91 03D0 1647 CMPB #MT$K_PE_1600,R0 ; See if user specified 1600.
    0A 13 03D3 1648 ASSUME MF$K_DENSITY_1600 EQ 0 ; Branch if yes, 1600 BPI.
    03D5 1649 BEQL 30$
    00BA C5 96 03D5 1650 ASSUME MF$K_DENSITY_6250 EQ 1 ; Else indicate 6250 BPI.
    05 F0 03D9 1651 INCB UCBSB_MF_DENSITY(R5) ; And insure that UCBSL_DEVDEPEND winds
    08 03DB 1652 INSV #MT$K_GCR_6250,- ; up with the correct value for DENSITY
    05 03DC 1653 #MT$V_DENSITY,- ; even though we may have arrived here
    7C A5 03DD 1654 UCBSW_BOFF(R5) ; by default.
    03DF 1655
    03F 1656 30$:
    51 F0 03DF 1658 INSV R1,-
    03E1 1659 #MF_TC_V_FMT,-
    03E1 1660 #MF_TC_S_FMT,-
    03E3 1661 UCBSW_MF_TMPLTC(R5) ; Set new FORMAT in UCB.
    44 A5 52 AA 03E6 1662 BICW R2,UCBSL_DEVDEPEND(R5) ; Use mask to clear out fields.
    52 52 D2 03EA 1663 MCOML R2,R2 ; Reverse mask for setting.
    7C A5 52 AA 03ED 1664 BICW R2,UCBSW_BOFF(R5) ; Clear unwanted user specified parameters.
    7C A5 A8 03F1 1665 BISW UCBSW_BOFF(R5),-
    44 A5 03F4 1666 UCBSL_DEVDEPEND(R5) ; Reset the verified parameters
    50 01 3C 03F6 1667 MOVZWL S^#SS$_NORMAL,R0 ; Indicate success.
    03F9 1668 SETMODE_EXIT:
    0F96 31 03F9 1669 BRW FUNCTION_EXIT ; Branch to common exit.
```

```
03FC 1671      .SBTTL Start a SKIPFILE or a SPACEFILE function
03FC 1672
03FC 1673      : START_SKIPFILE and START_SPACEFILE
03FC 1674      :
03FC 1675      : INPUTS:
03FC 1676      :     IRP$M_MEDIA = # files to skip (word count in longword field)
03FC 1677      :     R3 => TM78 CSR
03FC 1678      :     R5 => UCB
03FC 1679      :
03FC 1680
03FC 1681      START_SKIPFILE:      ; Logical I/O operation.
03FC 1682      START_SPACEFILE:    ; Physical I/O operation.
03FC 1683      MOVL      UCB$M_IRP(R5),R2      ; R2 => current IRP.
52  58 A5 D0 0400 1684      CVTWL      IRP$M_MEDIA(R2),R1      ; Pickup number of files to skip.
51  38 A2 32 0404 1685      BWLSS      REV_SPACEFILE      ; Branch if reverse skip.
0409 1686
7C  A5  51 B0 0409 1687      MOVW      R1,UCB$W_BOFF(R5)      ; Use convenient place in UCB to retain
040D 1688      ; number of files left to skip.
7E  A5  51 B0 040D 1689      MOVW      R1,UCB$W_BCNT(R5)      ; Use convenient place in UCB to keep
0411 1690      ; original number of files to skip.
0411 1691
0411 1692      :
0411 1693      : Here we loop skipping records at a clip of 255 per skip since the repeat
0411 1694      : count in the hardware command register is only 8 bits long. After
0411 1695      : each TAPEMARK is encountered we subtract 1 from UCB$W_BOFF until it
0411 1696      : reaches zero. Meanwhile we keep UCB$M_RECORD uptodate with the
0411 1697      : current tape position.
0411 1698      :
0411 1699
0411 1700      SKIP_FF_LOOP:      ; SKIP FILE FORWARD LOOP.
7C  A5  B5 0411 1701      TSTW      UCB$W_BOFF(R5)      ; See if any more files left to skip.
0414 1702      BWEQL      SKIP_NORMAL_EXIT      ; EQL implies all done, so branch.
0419 1703
0419 1704      EX_NDT_CMD      CMD=SP_FWD_REC,-      ; Execute space forward record
0419 1705      REPEAT=#255,-      ; NDT command to skip 255 records
0419 1706      TIMEOUT=#TU78 MAX SPACE,-; and timeout in N seconds and
0419 1707      ERROR_LABEL=SKIP_EXIT ; branch to SKIP_EXIT if we fail.
0432 1708
0432 1709      :
0432 1710      : Test interrupt code. Remember EX_NDT_CMD returns the interrupt code in R0.
0432 1711      :
0432 1712
50  01  91 0432 1713      CMPB      #HIC_DONE,R0      ; See if the command DONE.
0435 1714      BNEQ      10$      ; If not, branch around for more tests.
0437 1715
0437 1716      : If HIC_DONE then,
0437 1717
50  00BD C5 9A 0437 1718      MOVZBL      UCB$W_MF_NDTCR+1(R5),R0 ; Pickup number of RECORDS skipped.
00B0 C5 50 C0 043C 1719      ADDL      R0,UCB$M_RECORD(R5) ; Add in to maintain tape position.
FFCD 31 0441 1720      BRW      SKIP_FF_LOOP      ; And loop back to continue skipping.
0444 1721
50  02  91 0444 1722 10$:      CMPB      #HIC_TM,R0      ; See if TAPEMARK encountered.
0447 1723      BWNEQ      SKIP_ERROR      ; If not, branch around for more tests
044C 1724      ; to determine exact error condition.
044C 1725
044C 1726      : If HIC_TM then,
044C 1727
```



```
023A 30 044C 1728      BSBW  COMPUTE_NDT_REPEAT      ; R1 = # records skipped before
                                044F 1729                ; TAPEMARK encountered.
                                044F 1730
00B0 C5 51 D6 044F 1731      INCL  R1                ; Add in TAPEMARK.
                                C0 0451 1732      ADDL  R1,UCB$L_RECORD(R5)      ; Update tape position.
                                7C A5 B7 0456 1733      DECW  UCB$W_BOFF(R5)      ; Decrement # files left to skip.
                                00C8 C5 C3 0459 1734      SUBL3  UCB$L_MF_PREVTM(R5),-
50 00B0 C5 045D 1735      UCB$L_RECORD(R5),R0      ; R0 = distance between last 2 TAPEMARKS.
                                13 E1 0461 1736      BBC  #DEV$V_MNT,-          ; If NOT mounted then go
                                05 38 A5 0463 1737      UCB$L_DEVCHAR(R5),20$      ; test for possible EOF.
                                18 E1 0466 1738      BBC  #DEV$V_FOR,-          ; If mounted NOT foreign, (i.e. ACP NOT
0E 38 A5 0468 1739      UCB$L_DEVCHAR(R5),30$      ; NOT involved) branch around test EOF.
                                50 D7 046B 1740 20$:  DECL  R0                  ; See if last 2 TAPEMARKs adjacent.
                                046D 1741      BWEQL  SKIP_FILE_SETEOV      ; EQL => yes so branch.
                                0472 1742
                                0472 1743 ; We are here if either the volume is mounted, NOT foreign (i.e. ANSI mounted
                                0472 1744 ; with the ACP involved) and/or the last two TAPEMARKs are NOT adjacent.
                                0472 1745
00B0 C5 D0 0472 1746      MOVL  UCB$L_RECORD(R5),-      ; If NOT between tapemarks on /FOR tape,
00C8 C5 0476 1747      UCB$L_MF_PREVTM(R5)          ; remember position of this TAPEMARK.
FF95 31 0479 1748 30$:  BRW  SKIP_FF_LOOP          ; Branch back to skip some more.
```

```
047C 1750 : REV_SPACEFILE - reverse spacefile.
047C 1751 :
047C 1752 : R1 = negative of number of files to skip.
047C 1753 : R3 => TM78 CSR
047C 1754 : R5 => UCB
047C 1755 :
047C 1756 :
047C 1757 REV_SPACEFILE:
7C A5 51 AE 047C 1758 MNEGW R1,UCBSW_BOFF(R5) ; Use convenient place in UCB to retain
0480 1759 ; the number of files left to skip.
7E A5 51 AE 0480 1760 MNEGW R1,UCBSW_BCNT(R5) ; Use convenient place in UCB to keep
0484 1761 ; original number of files to skip.
0484 1762 :
0484 1763 :
0484 1764 : Here we loop skipping records in reverse at a clip of 255 records per skip
0484 1765 : since the repeat count in the hardware command register is only 8
0484 1766 : bits long. After each TAPEMARK is encountered we subtract 1 from
0484 1767 : UCB$W_BOFF until it reaches zero. Meanwhile we keep UCB$L_RECORD
0484 1768 : up to date with the current tape position. BOT ends the operation
0484 1769 : prematurely.
0484 1770 :
0484 1771 :
0484 1772 SKIP_FR_LOOP: ; SKIP FILE REVERSE LOOP
7C A5 B5 0484 1773 TSTW UCB$W_BOFF(R5) ; See if done.
0487 1774 BWEQL SKIP_NORMAL_EXIT ; EQL implies done, so branch end.
048C 1775 :
048C 1776 EX_NDT_CMD CMD=SP_REV_REC,- ; Execute space reverse record
048C 1777 REPEAT=#255,- ; NDT command with a REPEAT of
048C 1778 TIMEOUT=#TU78 MAX SPACE,-; 255 and a TIMEOUT of N secs.
048C 1779 ERROR_LABEL=SKIP_EXIT ; Branch to SKIP_EXIT if we fail.
04A5 1780 :
04A5 1781 :
04A5 1782 : Test interrupt code which EX_NDT_CMD returns in R0.
04A5 1783 :
04A5 1784 :
50 01 91 04A5 1785 CMPB #HIC_DONE,R0 ; DONE implies that we skipped 255 recs.
0D 12 04A8 1786 BNEQ 10$ ; If not, branch for further tests.
04AA 1787 :
04AA 1788 : If HIC_DONE then,
04AA 1789 :
50 00BD C5 9A 04AA 1790 MOVZBL UCB$W_MF_NDTCR+1(R5),R0 ; R0 = original repeat count (255).
00B0 C5 50 C2 04AF 1791 SUBL R0,UCB$L_RECORD(R5) ; Subtract to update tape position.
FFCD 31 0484 1792 BRW SKIP_FR_LOOP ; Loop back to continue skipping.
0487 1793 :
0487 1794 10$:
50 02 91 0487 1795 CMPB #HIC_TM,R0 ; Did we encounter a TAPEMARK?
08 13 048A 1796 BEQL 20$ ; EQL implies yes.
50 03 91 048C 1797 CMPB #HIC_BOT,R0 ; Did we encounter BOT?
048F 1798 BWEQ SKIP_ERROR ; NEQ implies no; so branch around.
04C4 1799 :
04C4 1800 : If HIC_TM or HIC_BOT then,
04C4 1801 :
04C4 1802 20$:
01C2 30 04C4 1803 BSBW COMPUTE_NDT_REPEAT ; R1 = # of records actually skipped.
04C7 1804 :
04C7 1805 :
50 03 91 04C7 1806 CMPB #HIC_BOT,R0 ; Are we at BOT or TAPEMARK?
```

	05	13	04CA	1807	BEQL	30\$	; EQL implies BOT, so branch around.
	51	D6	04CC	1808	INCL	R1	; Bump count to include TAPEMARK skipped.
	7C	A5	B7	04CE	DECW	UCBSW_BOFF(R5)	; Decrement # files left to skip.
			04D1	1810			
				30\$:			
00B0	C5	51	C2	04D1	SUBL	R1,UCBSL_RECORD(R5)	; Update tape position.
00C8	C5	02	CE	04D6	MNEGL	#2,UCBSL_MF_PREVTM(R5)	; After reverse motion, set previous TM to -
	50	02	91	04DB	CMPB	#HIC_TM,R0	; Was it TAPEMARK?
				04DE	BWEQL	SKIP_FR_LOOP	; If so, branch back to continue skipping.
00B0	C5	D4	04E3	1815	CLRL	UCBSL_RECORD(R5)	; If at BOT, then should be zero.
	01	A8	04E7	1816	BISW	#<MTSM BOTa-16>,-	; Set BOT bit on in UCB field.
46	A5		04E9	1817		UCBSL_DEVDEPEND+2(R5)	
	10	AA	04EB	1818	BICW	#<MTSM LOSTa-16>,-	; Clear the fact that we may have
46	A5		04ED	1819		UCBSL_DEVDEPEND+2(R5)	; been lost
0186	31	04EF	1820	BRW		SKIP_NORMAL_EXIT	; If at BOT, then we are done.

```
04F2 1822      .SBTTL Start a SKIPRECORD or a SPACERECORD function
04F2 1823
04F2 1824      : START_SKIPRECORD and START_SPACERECORD
04F2 1825      :
04F2 1826      : INPUTS:
04F2 1827      :   IRP$L_MEDIA = # records to skip (word count in longword field)
04F2 1828      :   R3 => TM78 CSR
04F2 1829      :   R5 => UCB
04F2 1830      :
04F2 1831
04F2 1832      START_SKIPRECORD:
04F2 1833      START_SPACERECORD:
04F2 1834      MOVL    UCB$L_IRP(R5),R2      ; Logical I/O operation.
04F6 1835      CVTWL   IRP$L_MEDIA(R2),R1    ; Physical I/O operation.
04FA 1836      BWLSS   REV_SPACERECORD      ; R2 => current IRP.
04FF 1837      ;
04FF 1838      MOVW    R1,UCB$W_BOFF(R5)      ; Pickup number of records to skip.
0503 1839      ;
0503 1840      MOVW    R1,UCB$W_BCNT(R5)      ; Branch if reverse skip.
0507 1841      ;
0507 1842      ;
0507 1843      ;
0507 1844      : Here we loop skipping records at a clip of 255 or UCB$W_BOFF
0507 1845      : per skip since the repeat count in the hardware command
0507 1846      : register is only 8 bits long. After each HIC_DONE interrupt,
0507 1847      : we subtract the number of records skipped from UCB$W_BOFF
0507 1848      : until we are done.
0507 1849      :
0507 1850
0507 1851      SKIP_RF_LOOP:
0507 1852      TSTW    UCB$W_BOFF(R5)      ; SKIP RECORD FORWARD LOOP.
050A 1853      BWEQL   SKIP_NORMAL_EXIT    ; See if any more records left to skip.
050F 1854      ;
050F 1855      MOVZBL  #255,R0            ; EQL implies all done, so branch.
0513 1856      CMPW    R0,UCB$W_BOFF(R5)    ; Calculate repeat count, 255 or
0517 1857      BLSS    10$                ; UCB$W_BOFF, whichever is less.
0519 1858      MOVZWL  UCB$W_BOFF(R5),R0    ; LSS => 255 if less.
051D 1859      10$:
051D 1860      EX_NDT_CMD      CMD=SP_FWD_REC,- ; Else UCB$W_BOFF is less.
051D 1861      REPEAT=R0,-      ; Execute space forward record
051D 1862      TIMEOUT=#TU78 MAX SPACE,- ; NDT command to skip records
051D 1863      ERROR_LABEL=SKIP_EXIT ; and timeout in N seconds and
0535 1864      ; branch to SKIP_EXIT if we fail.
0535 1865      :
0535 1866      : Test interrupt code. Remember EX_NDT_CMD returns the interrupt code in R0.
0535 1867      :
0535 1868      :
0535 1869      CMPB    #HIC_DONE,R0      ; See if the command DONE.
0538 1870      BNEQ    20$            ; If not, branch around for more tests.
053A 1871      ;
053A 1872      : If HIC_DONE then,
053A 1873      :
053A 1874      MOVZBL  UCB$W_MF_NDTCR+1(R5),R0 ; Pickup number of RECORDS skipped.
053F 1875      ADDL    R0,UCB$L_RECORD(R5)    ; Add in to maintain tape position.
0544 1876      SUBW    R0,UCB$W_BOFF(R5)    ; Subtract # of records just skipped.
0548 1877      BRW     SKIP_RF_LOOP          ; And loop back to continue skipping.
054B 1878
```

52 58 A5 D0
51 38 A2 32
7C A5 51 B0
7E A5 51 B0
7C A5 B5
50 FF 8F 9A
7C A5 50 B1
50 7C A5 3C
50 01 91
11 12
50 00BD C5 9A
00B0 C5 50 C0
7C A5 50 A2
FFBC 31

```
50 02 91 054B 1879 20$: CMPB #HIC_TM,R0 ; See if TAPEMARK encountered.
      054E 1880 BwNEQ SKIP_ERROR ; If not, branch around for more tests
      0553 1881 ; to determine exact error condition.
      0553 1882
      0553 1883 ; If HIC_TM then,
      0553 1884
      0133 30 0553 1885 BSBW COMPUTE_NDT_REPEAT ; R1 = # records skipped before
      0556 1886 ; TAPEMARK encountered.
      51 D6 0556 1887 INCL R1 ; Add in TAPEMARK.
      0558 1888
      7C A5 51 A2 0558 1889 SUBW R1,UCB$W_BOFF(R5) ; Decrement # records left to skip.
      00B0 C5 51 C0 055C 1890 ADDL R1,UCB$L_RECORD(R5) ; Update tape position.
      00C8 C5 C3 0561 1891 SUBL3 UCB$L_MF_PREVTM(R5),-
      50 00B0 C5 0565 1892 UCB$L_RECORD(R5),R0 ; R0 = distance between last 2 TAPEMARKS.
      13 E1 0569 1893 BBC #DEV$V_MNT,- ; If NOT mounted then go
      05 38 A5 056B 1894 UCB$L_DEVCHAR(R5),30$ ; test for possible EOv.
      18 E1 056E 1895 BBC #DEV$V_FOR,- ; If mounted NOT foreign, (i.e. ACP NOT
      0E 38 A5 0570 1896 UCB$L_DEVCHAR(R5),40$ ; NOT involved) branch around test EOv.
      50 D7 0573 1897 30$: DECL R0 ; See if last 2 TAPEMARKs adjacent.
      0575 1898 BWEQL SKIP_RECORD_SETEOV ; EQL => yes so branch.
      057A 1899
      057A 1900 ; We are here if either the volume is mounted, NOT foreign (i.e. ANJl mounted
      057A 1901 ; with the ACP involved) and/or the last two TAPEMARKs are NOT adjacent.
      057A 1902
      00B0 C5 D0 057A 1903 MOVL UCB$L_RECORD(R5),- ; If NOT between tapemarks on /FOR tape,
      00C8 C5 057E 1904 UCB$L_MF_PREVTM(R5) ; remember position of this TAPEMARK.
      00E9 31 0581 1905 40$: BRW SKIP_SETEOF ; Branch to set status and exit.
```

```
0584 1907 : REV_SPACERECORD - reverse spacer record.
0584 1908 :
0584 1909 : R1 = negative of number of records to skip.
0584 1910 : R3 => TM78 CSR
0584 1911 : R5 => UCB
0584 1912 :
0584 1913 :
0584 1914 REV_SPACERECORD:
7C A5 51 AE 0584 1915 MNEGW R1,UCBSW_BOFF(R5) ; Use convenient place in UCB to retain
0588 1916 ; the number of records left to skip.
7E A5 51 AE 0588 1917 MNEGW R1,UCBSW_BCNT(R5) ; Use convenient place in UCB to keep
058C 1918 ; original number of records to skip.
058C 1919 :
058C 1920 : Here we loop skipping records in reverse at a clip of 255 or UCB$W_BOFF
058C 1921 : per skip since the repeat count in the hardware command
058C 1922 : register is only 8 bits long. After each HIC_DONE interrupt,
058C 1923 : we subtract the number of records skipped from UCB$W_BOFF
058C 1924 : until we are done.
058C 1925 :
058C 1926 :
058C 1927 SKIP_RR_LOOP: ; SKIP RECORD REVERSE LOOP.
7C A5 B5 058C 1928 TSTW UCB$W_BOFF(R5) ; See if any more records left to skip.
058F 1929 BWEQL SKIP_NORMAL_EXIT ; EQL implies all done, so branch.
0594 1930
50 FF 8F 9A 0594 1931 MOVZBL #255,R0 ; Calculate repeat count, 255 or
7C A5 50 B1 0598 1932 CMPW R0,UCBSW_BOFF(R5) ; UCB$W_BOFF, whichever is less.
059C 1933 BLSS 10$ ; LSS => 255 if less.
50 7C A5 3C 059E 1934 MOVZWL UCB$W_BOFF(R5),R0 ; Else UCB$W_BOFF is less.
05A2 1935 10$:
05A2 1936 EX_NDT_CMD CMD=SP_REV_REC,- ; Execute space reverse record
05A2 1937 REPEAT=R0,- ; NDT command to skip records
05A2 1938 TIMEOUT=#TU78_MAX_SPACE,- ; and timeout in N seconds and
05A2 1939 ERROR_LABEL=SKIP_EXIT ; go to SKIP_EXIT if we fail.
05BA 1940
05BA 1941 :
05BA 1942 : Test interrupt code. Remember EX_NDT_CMD returns the interrupt code in R0.
05BA 1943 :
05BA 1944 :
50 01 91 05BA 1945 CMPB #HIC_DONE,R0 ; See if the command DONE.
11 12 05BD 1946 BNEQ 20$ ; If not, branch around for more tests.
05BF 1947
05BF 1948 : If HIC_DONE then,
05BF 1949
50 00BD C5 9A 05BF 1950 MOVZBL UCB$W_MF_NDTCR+1(R5),R0 ; Pickup number of RECORDS skipped.
00B0 C5 50 C2 05C4 1951 SUBL R0,UCBSL_RECORD(R5) ; Subtract to maintain tape position.
7C A5 50 A2 05C9 1952 SUBW R0,UCBSW_BOFF(R5) ; Subtract # of records just skipped.
FFBC 31 05CD 1953 BRW SKIP_RR_LOOP ; And loop back to continue skipping.
05D0 1954
50 02 91 05D0 1955 20$: CMPB #HIC_TM,R0 ; See if TAPEMARK encountered.
08 13 05D3 1956 BEQL 30$ ; EQL implies yes.
50 03 91 05D5 1957 CMPB #HIC_BOT,R0 ; Did we encounter BOT?
05D8 1958 BWEQ SKIP_ERROR ; NEQ implies no; so branch around.
05DD 1959
05DD 1960 : If HIC_TM or HIC_BOT then,
05DD 1961
05DD 1962 30$:
00A9 30 05DD 1963 BSBW COMPUTE_NDT_REPEAT ; R1 = # of records actually skipped.
```

50	03	91	05E0	1964				
	02	13	05E3	1966				
	51	D6	05E5	1967				
			05E7	1968	40\$:			
7C	A5	51	A2	05E7	1969			
00B0	C5	51	C2	05EB	1970			
00C8	C5	02	CE	05F0	1971			
	50	02	91	05F5	1972			
				05F8	1973			
00B0	C5	D4	05FD	1974				
	01	A8	0601	1975				
46	A5		0603	1976				
	10	AA	0605	1977				
46	A5		0607	1978				
0061		31	0609	1979				

CMPB	#HIC_BOT,R0	; Are we at BOT or TAPEMARK?
BEQL	40\$	; EQL implies BOT, so branch around.
INCL	R1	; Bump count to include TAPEMARK skipped.
SUBW	R1,UCBSW_BOFF(R5)	; Decrement # records left to skip.
SUBL	R1,UCBSL_RECORD(R5)	; Update tape position.
MNEGL	#2,UCBSL_MF_PREVTM(R5)	; After reverse motion, set previous TM to -
CMPB	#HIC_TM,R0	; Was it TAPEMARK?
BWEQL	SKIP_SETEOF	; If so, branch around to exit.
CLRL	UCBSL_RECORD(R5)	; If at BOT, then should be zero.
BISW	#<MTSM_BOT@-16>,-	; Set BOT bit on in UCB field.
BICW	#<MTSM_LOST@-16>,-	; Clear the fact that we may have
	UCBSL_DEVDEPEND+2(R5)	; been lost
BRW	SKIP_SETEOF	; If at BOT, then we are all done.

```
060C 1981 ; SKIP EXIT AND ERROR PROCESSING.
060C 1982
060C 1983 SKIP_FILE SETEOV:
060C 1984 SKIP_RECORD SETEOV:
7C A5 B6 060C 1985 INCW UCBSW_BOFF(R5) ; Increment to indicate 1 less file skipped.
060F 1986 EX_NDT_CMD CMD=SP_REV_REC,- ; Execute space REVERSE record
060F 1987 REPEAT=#1,- ; NDT command to skip back over
060F 1988 TIMEOUT=#MINIMUM_TIMEOUT,- ; last of 2 adjacent TAPEMARKs.
060F 1989 ERROR_LABEL=SKIP_EXIT ; Branch to SKIP_EXIT if we fail.
0623 1990
0623 1991 ; EX_NDT_CMD returns interrupt code in R0.
0623 1992
50 02 91 0623 1993 CMPB #HIC_TM,R0 ; It had better be a TAPEMARK.
23 12 0626 1994 BNEQ SKIP_HARD_ERROR ; NEQ implies some error.
0628 1995
50 00B0 C5 D7 0628 1996 DECL UCBSL_RECORD(R5) ; Decrement to record backward movement.
09A0 8F 3C 062C 1997 MOVZWL #SS$_ENDOFVOLUME,R0 ; Set final I/O status.
48 11 0631 1998 BRB SKIP_EXIT ; Branch around to exit.
0633 1999
0633 2000 SKIP_ERROR:
50 0D 91 0633 2001 CMPB #HIC_NOT_CAPABLE,R0 ; See if blank tape.
07 12 0636 2002 BNEQ 10$ ; NEQ means no, not blank tape.
50 02D4 8F 3C 0638 2003 MOVZWL #SS$_OPINCOMPL,R0 ; Else get proper error status and
3C 11 063D 2004 BRB SKIP_EXIT ; branch around to end.
063F 2005
50 0B 91 063F 2006 10$: CMPB #HIC_OFF_LINE,R0 ; See if OFF-LINE.
07 12 0642 2007 BNEQ SKIP_HARD_ERROR ; If not, branch around to test further.
50 01A4 8F 3C 0644 2008 MOVZWL #SS$_MEDOFL,R0 ; Indicate media off-line and
30 11 0649 2009 BRB SKIP_EXIT ; branch around to end.
064B 2010
064B 2011 SKIP_HARD_ERROR:
0B7F 30 064B 2012 BSBW DEVICE_ERROR ; Do EXTENDED SENSE and log device error
064E 2013
50 18 91 064E 2014 CMPB #HIC_TM_FAULT_A,R0 ; See if controller error.
07 12 0651 2015 BNEQ 20$ ; If not, branch around to next test.
50 0054 8F 3C 0653 2016 MOVZWL #SS$_CTRLERR,R0 ; Indicate controller error and
21 11 0658 2017 BRB SKIP_EXIT ; branch around.
065A 2018
50 0C 91 065A 2019 20$: CMPB #HIC_NON_EX,R0 ; Test if non-existent drive.
07 12 065D 2020 BNEQ 30$ ; If not then branch to drive error.
50 01C4 8F 3C 065F 2021 MOVZWL #SS$_NONEXDRV,R0 ; Indicate non-existent drive and
15 11 0664 2022 BRB SKIP_EXIT ; branch around to exit.
0666 2023
50 008C 8F 3C 0666 2024 30$: MOVZWL #SS$_DRVERR,R0 ; Indicate drive error
0E 11 066B 2025 BRB SKIP_EXIT ; Branch around to end.
066D 2026 SKIP_SETEOF:
50 0870 8F 3C 066D 2027 MOVZWL #SS$_ENDOFFILE,R0 ; Indicate status return.
46 A5 02 A8 0672 2028 BISW2 #<MTSM_EOF@-16>,UCBSL_DEVDEPEND+2(R5) ; Set EOF in device depend
03 11 0676 2029 BRB SKIP_EXIT ; And branch around to exit.
0678 2030 SKIP_NORMAL_EXIT:
50 01 9A 0678 2031 MOVZBL S^#SS$_NORMAL,R0 ; Indicate normal status.
067B 2032 SKIP_EXIT:
7C A5 A3 067B 2033 SUBW3 UCBSW_BOFF(R5),- ; Calculate the number of files
7E A5 067E 2034 UCBSW_BCNT(R5),R1 ; skipped.
50 10 10 51 F0 0681 2035 INSV R1,#16,#16,R0 ; And place count in high word of R0.
0D09 31 0686 2036 BRW FUNCTION_EXIT ; Branch to common exit.
```



```
0689 2038 .SBTTL COMPUTE_NDT_REPEAT subroutine
0689 2039
0689 2040 ; COMPUTE_NDT_REPEAT - subroutine to compute the actual number of times a
0689 2041 ; Non-data transfer (NDT) command has been repeated.
0689 2042 ;
0689 2043 ; INPUTS:
0689 2044 ; R3 => TM78 CSR
0689 2045 ; R5 => UCB
0689 2046 ; UCBSW_MF_NDTCR contains a copy of original NDT
0689 2047 ; Command loaded into unit's NDT register.
0689 2048 ;
0689 2049 ; OUTPUT:
0689 2050 ; R1 = actual number of times the NDT command was repeated.
0689 2051 ;
0689 2052 ; Subroutine extracts original repeat count from UCBSW_MF_NDTCR and
0689 2053 ; subtracts residue count from device unit's NDT command register.
0689 2054 ;
0689 2055 ;
0689 2056 COMPUTE_NDT_REPEAT:
0689 2057
51 54 A5 3C 0689 2058 MOVZWL UCBSW_UNIT(R5),R1 ; Get this device's unit number.
30 A341 DD 068D 2059 PUSHL MF_NDT0(R3)[R1] ; Push residue of previous NDT command.
08 EF 0691 2060 EXTZV #MF_NDT0_V_CCNT,- ; Extract residual repeat count and
6E 08 0693 2061 #MF_NDT0_S_CCNT,(SP),- ; zero extend it and then leave
6E 0695 2062 (SP) ; on the top of the stack.
51 00BD C5 9A 0696 2063 MOVZBL UCBSW_MF_NDTCR+1(R5),R1 ; Pickup original repeat count.
51 8E C2 069B 2064 SUBL (SP)+,R1 ; Subtract to get # completed.
05 069E 2065 RSB ; Return to caller.
```

```
069F 2067      .SBTTL Start WRITEOF or WRITEMARK or DSE function
069F 2068
069F 2069      ; START_WRITEOF and START_WRITEMARK and START_DSE
069F 2070      ;
069F 2071      ; INPUTS:
069F 2072      ;      R3 => TM78 CSR
069F 2073      ;      R5 => UCB
069F 2074      ;
069F 2075
069F 2076      START_WRITEOF:      ; Logical I/O function.
069F 2077      START_WRITEMARK:    ; Physical I/O function.
069F 2078      MOVL      #MINIMUM_TIMEOUT,-      ;
069F 2079      UCB$M MF_TIMEOUT(R5)      ; Indicate TIMEOUT for next NDT operation.
069F 2080      MOVZBL    #F_WTM_GCR!GO_BIT,R0      ; Assume 6250 density
069F 2081      TSTB      UCB$B MF_DENSITY(R5)      ; Test the density.
069F 2082      BNEQ      GO_WRITEMARK_OR_ERASE      ; NEQ implies 6250, so branch around.
069F 2083      MOVZBL    #F_WTM_PE!GO_BIT,R0      ; If 1600, set proper command code
069F 2084      BRB       GO_WRITEMARK_OR_ERASE      ; Join common WRITE_TAPEMARK/ERASETAPE code.
069F 2085      START_ERASETAPE:
069F 2086      MOVL      #MINIMUM_TIMEOUT,-      ;
069F 2087      UCB$M MF_TIMEOUT(R5)      ; Indicate TIMEOUT for next NDT operation.
069F 2088      MOVZBL    #F_ERG_GCR!GO_BIT,R0      ; Assume 6250 density.
069F 2089      TSTB      UCB$B MF_DENSITY(R5)      ; Test the density.
069F 2090      BNEQ      GO_WRITEMARK_OR_ERASE      ; NEQ implies 6250, so branch around.
069F 2091      MOVZBL    #F_ERG_PE!GO_BIT,R0      ; If 1600, set proper command code.
069F 2092      BRB       GO_WRITEMARK_OR_ERASE      ; Join common WRITE_TAPEMARK/ERASETAPE code.
069F 2093      START_DSE:
069F 2094      MOVL      #TU78_MAX_DSE,UCB$M MF_TIMEOUT(R5)      ;
069F 2095      ; Indicate TIMEOUT for next NDT operation.
069F 2096      MOVZBL    #F_DSE!GO_BIT,R0      ; Set NDT command in R0.
069F 2097      GO_WRITEMARK_OR_ERASE:      ; COMMON CODE.
069F 2098      EX_NDT_CMD      CMD=R0,-      ; Execute write tape mark or erase command
069F 2099      ERROR_LABEL=WRITEMARK_EXIT
069F 2100
069F 2101      ;
069F 2102      ; EX_NDT CMD returns the interrupt code in R0. After a WRITE-TAPE-MARK or
069F 2103      ; ERASE command the following interrupt codes are possible and the
069F 2104      ; following actions are taken:
069F 2105      ;
069F 2106      ; 1. HIC_DONE - indicates that the operation terminated successfully.
069F 2107      ;      We simply increment the tape position counter (UCB$M_RECORD)
069F 2108      ;      and return SSS_NORMAL status.
069F 2109      ;
069F 2110      ; 2. HIC_EOT - indicates that the operation terminated successfully but
069F 2111      ;      the tape is now positioned beyond the reflective strip at the
069F 2112      ;      end of the tape. We increment the tape position counter
069F 2113      ;      (UCB$M_RECORD) and return SSS_ENDOFTAPE status. Also
069F 2114      ;      we set the MTS$M_EOT bit in UCB$M_DEVDEPEND.
069F 2115      ;
069F 2116      ; 3. HIC_FPT - we could not write the TAPEMARK due to the fact that the
069F 2117      ;      tape was WRITE-LOCKED. We return SSS_WRTILCK status.
069F 2118      ;
069F 2119      ; 4. HIC_OFF_LINE - indicates that the drive is offline and therefore we
069F 2120      ;      were unable to complete the operation. We return SSS_MEDOFL
069F 2121      ;      status.
069F 2122      ;
069F 2123      ; 5. HIC_NON_EX - indicates that the HARDWARE device is non-existent and
069F 2124      ;      therefore we were unable to complete the operation. We return
069F 2125      ;      SSS_NONEXDRV status.
069F 2126      ;
069F 2127      ; 6. HIC_TM_FAULT_A - indicates the TM78 controller experienced an error
069F 2128      ;      that prevented successful completion of the operation. We
```

```
06DB 2124 : return SSS_CTLERR status.
06DB 2125 : 7. HIC_NOT_AVAIL, HIC_BAD_TAPE, HIC_TU_FAULT_A - all indicate that
06DB 2126 : a hardware error prevented successful termination of the I/O
06DB 2127 : function. We return SSS_DRVERR status.
06DB 2128 :
06DB 2129 :
50 01 91 06DB 2130 CMPB #HIC_DONE,R0 ; See if successful.
10 13 06DE 2131 BEQL 10$ ; EQL implies yes.
50 04 91 06E0 2132 CMPB #HIC_EOT,R0 ; See if successful but beyond EOT.
37 12 06E3 2133 BNEQ 40$ ; If not, branch around to next test.
50 0878 8F 3C 06E5 2134 MOVZWL #SS$ ENDOFTAPE,R0 ; Indicate final status.
04 A8 06EA 2135 BLSW #<MTSM EOT@-16>,- ; Set bit indicating we are
46 A5 06EC 2136 UCB$$_DEVDEPEND+2(R5) ; beyond EOT marker.
03 11 06EE 2137 BRB 20$ ; Branch around.
06F0 2138
50 01 3C 06F0 2139 10$: MOVZWL S^#SS$_NORMAL,R0 ; Indicate success.
20$: 06F3 2140
00 EF 06F3 2141 EXTZV #IRP$V_FCODE,- ; See which command we are executing.
06 06F5 2142 #IRP$$_FCODE,- ; If it is ERASETAPE, then branch
51 009A C5 06F6 2143 UCB$$_FUNC(R5),R1 ; around the incrementation of
06 51 D1 06FA 2144 CMPL R1,#10$_ERASETAPE ; UCB$$_RECORD and the setting of
1B 13 06FD 2145 BEQL 30$ ; UCB$$_MF_PREVTM.
15 51 D1 06FF 2146 CMPL R1,#10$_DSE ; If it is NOT 10$_DSE, branch around
08 12 0702 2147 BNEQ 25$ ; the clearing of UCB$$_RECORD and
00B0 C5 D4 0704 2148 CLRL UCB$$_RECORD(R5) ; the initializing of UCB$$_MF_PREVTM
00C8 C5 02 CE 0708 2149 MNEGL #2,UCB$$_MF_PREVTM(R5) ; to the value negative 2 (-2).
08 11 070D 2150 BRB 30$ ; Branch around.
070F 2151 25$:
00B0 C5 D6 070F 2152 INCL UCB$$_RECORD(R5) ; Increment position counter.
00B0 C5 D0 0713 2153 MOVL UCB$$_RECORD(R5),- ; Remember this as position of previous
00C8 C5 0717 2154 UCB$$_MF_PREVTM(R5) ; TAPEMARK which we have just written.
38 11 071A 2155 30$: BRB WRITEMARK_EXIT ; Branch around to exit.
071C 2156
50 08 91 071C 2157 40$: CMPB #HIC_FPT,R0 ; Was the problem lack of RING.
07 12 071F 2158 BNEQ 50$ ; NEQ implies no, so branch to next test.
50 025C 8F 3C 0721 2159 MOVZWL #SS$ WRITLCK,R0 ; Indicate WRITELOCK status and
2C 11 0726 2160 BRB WRITEMARK_EXIT ; branch around to exit.
0728 2161
50 08 91 0728 2162 50$: CMPB #HIC_OFF_LINE,R0 ; Is drive offline?
07 12 072B 2163 BNEQ 60$ ; NEQ implies no, so branch to next test.
50 01A4 8F 3C 072D 2164 MOVZWL #SS$ MEDOFL,R0 ; Indicate final status and
20 11 0732 2165 BRB WRITEMARK_EXIT ; branch around to exit.
0734 2166 60$:
0A96 30 0734 2167 BSBW DEVICE_ERROR ; Do EXTENDED SENSE and log device error
0737 2168
50 18 91 0737 2169 CMPB #HIC_TM_FAULT_A,R0 ; Is problem, controller problem?
07 12 073A 2170 BNEQ 70$ ; If not, branch around to next test.
50 0054 8F 3C 073C 2171 MOVZWL #SS$ CTLERR,R0 ; Else indicate final status and
11 11 0741 2172 BRB WRITEMARK_EXIT ; branch around to exit.
0743 2173
50 0C 91 0743 2174 70$: CMPB #HIC_NON_EX,R0 ; Is problem, NON-existent drive?
07 12 0746 2175 BNEQ 80$ ; If not, branch around to next test.
50 01C4 8F 3C 0748 2176 MOVZWL #SS$ NONEXDRV,R0 ; Else indicate final status and
05 11 074D 2177 BRB WRITEMARK_EXIT ; branch around to exit.
074F 2178
50 008C 8F 3C 074F 2179 80$: MOVZWL #SS$_DRVERR,R0 ; Indicate DRIVE error.
0754 2180
```

TFDRIVER
V04-000

E 8
- TM78/TU78 MAGTAPE DRIVER
Start WRITEOF or WRITEMARK or DSE functi

16-SEP-1984 00:08:15 VAX/VMS Macro V04-00
5-SEP-1984 00:17:37 [DRIVER.SRC]TFDRIVER.MAR;1

Page 49
(1)

TFD
V04

0C3B 31 0754 2181 WRITEMARK_EXIT:
0754 2182 BRW FUNCTION_EXIT ; Branch to common exit.

```
0757 2184 .SBTTL NDT Function Executor.
0757 2185
0757 2186 : NDT_EXECUTOR - Subroutine called by code generated by EX_NDT_CMD macro.
0757 2187 : This subroutine performs the common part of the execution of all
0757 2188 : NDT commands.
0757 2189 :
0757 2190 : INPUTS:
0757 2191 : R3 => TM78 CSR
0757 2192 : R5 => UCB
0757 2193 : UCBSW_MF_NDTCR(R5) contains the command to put into the NDT register
0757 2194 : UCBSL_MF_TIMEOUT(R5) contains the TIMEOUT amount
0757 2195 :
0757 2196 : OUTPUTS:
0757 2197 : If we do NOT get timed cut, we return to the point of invocation plus 2
0757 2198 : and:
0757 2199 :
0757 2200 : R0 = HIC (Hardware Interrupt Code) returned by TM78
0757 2201 :
0757 2202 : If we get a TIMEOUT and experience return from POWERFAIL_TIMEOUT we
0757 2203 : return to the ERROR_LABEL whose relative address is stored
0757 2204 : at the point of invocation.
0757 2205 :
0757 2206 :
0757 2207 NDT_EXECUTOR:
0757 2208
0757 2209 SUBSAVE
150 00BC C5 3C 0761 2210 MOVZWL UCBSW_MF_NDTCR(R5),R0 ; Push invocation point onto SUBSTACK.
0180 C5 50 D0 0766 2211 MOVL R0,UCBSL_MF_CMD(R5) ; Record Hardware command about to
076B 2212 ; become the CURRENT command.
076B 2213 :
076B 2214 : For all but REWIND, UNLOAD, and DSE, we sequentialize the following so as
076B 2215 : not to software timeout due to activity on other drives.
076B 2216 : This is due to the fact that although the TM78 will allow us to queue up
076B 2217 : parallel requests it in fact will not process them in parallel and
076B 2218 : therefore activity on other drives can mean inordinate waits which
076B 2219 : would trigger software timeouts.
076B 2220 : Sequentialization is enforced by using the PRIMARY channel (i.e. the TM78
076B 2221 : IDB) as an interlock or MUTEX.
076B 2222 :
076B 2223 :
150 07 91 076B 2224 CMPB #F_REWIND!GO_BIT,R0 ; See if a REWIND.
150 10 13 076E 2225 BEQL 10$ ; If SO branch around and DON'T REQPCN
150 05 91 0770 2226 CMPB #F_UNLOAD!GO_BIT,R0 ; See if a UNLOAD.
150 0B 13 0773 2227 BEQL 10$ ; If SO branch around and DON'T REQPCN
150 0B 91 0775 2228 CMPB #F_DSE!GO_BIT,R0 ; See if a DSE.
150 06 13 0778 2229 BEQL 10$ ; If SO branch around and DON'T REQPCN
077A 2230 REQPCN ; Request TM78 channel for all others.
0780 2231 10$:
0780 2232 DSBINT
0786 2233 PRIOR_TEST TMRDY ; Assure that TM78 ready.
150 05 E0 078F 2234 BBS #UCBSV_POWER,- ; Branch if POWERFAIL.
59 64 A5 0791 2235 UCBSW_STS(R5),40$ ; R0 = unit number of device.
50 54 A5 3C 0794 2236 MOVZWL UCBSW_UNIT(R5),R0
00BC C5 3C 0798 2237 MOVZWL UCBSW_MF_NDTCR(R5),-
30 A340 079C 2238 MF_NDTCR(R3)[R0] ; Move command to proper register.
079F 2239 WFIKPCN 50$,UCBSL_MF_TIMEOUT(R5) ; Wait for interrupt.
07AB 2240 POST_TEST_TMRDY ; Assure that TM78 ready.
```

			07B4	2241				: Assume not REWINDING or LOADING.
			07B4	2242				:
	00	EF	07B4	2243	EXTZV	#MF_NDTA_V_NDIC,-		; Extract interrupt code from attention
	06		07B6	2244		#MF_NDTA_S_NDIC,-		; data saved in UCB by interrupt
52	0168 C5		07B7	2245		UCBSL_MF_NDTA(R5),R2		; dispatcher, and place in R2.
52	52	91	07BB	2246	CMPB	#HIC_NOT_RDY,R2		; See if NOT-READY.
			07BE	2247				; NOT READY implies the drive has been
			07BE	2248				; MANUALLY rewound or loaded.
	07	12	07BE	2249	BNEQ	20\$		; If NOT, branch around.
	01	A8	07C0	2250	BISW	#UCBSM MF_REWIND,-		; If in the middle of MANUAL REWIND,
68	A5		07C2	2251		UCBSW_DEVSTS(R5)		; then set appropriate bit.
	OB72	31	07C4	2252	BRW	AWAIT_MANUAL_REWIND		; If NOT-READY, branch to wait for
			07C7	2253				; rewind or load to finish before
			07C7	2254				; trying to restart the I/O operation.
			07C7	2255	20\$: CMPB	#HIC_REWINDING,R2		; See if REWINDING interrupt.
52	07	91	07C7	2256				:
	04	12	07CA	2257	BNEQ	30\$		; If not, branch around.
	01	A8	07CC	2258	BISW	#UCBSM MF_REWIND,-		; If in the middle of REWIND, then
68	A5		07CE	2259		UCBSW_DEVSTS(R5)		; set appropriate bit.
			07D0	2260	30\$: MOVL	UCBSL_MF_NDTA(R5)-		; Copy NDI attention data to a
			07D0	2261		UCBSL_MF_NDTA_C(R5)		; stable place in the UCB.
0168	C5	D0	07D0	2262				:
0184	C5		07D4	2263	BSBW	SAVE_TM7B_REGS		; Save hardware registers for ERROR LOG
	OF74	30	07D7	2264	IOfORK			; Lower IPL to FORK level.
			07DA	2265	BBS	#UCBSV_POWER,-		:
	05	E0	07E0	2266		UCBSW_STS(R5),50\$		; Branch if we had a POWERFAIL.
OB	64 A5		07E2	2267				; Release channel in case we have it.
			07E5	2268	RELCHAN			; Branch around if not.
			07EB	2269	BRB	60\$		; We had a POWERFAIL.
			07ED	2270	ENBINT			; or a TIMEOUT.
			07FO	2271	SETIPL	UCBSB_FIPL(R5)		; Call routine.
	067F	30	07F4	2272	BSBW	TIMOUT_POWERFAIL		; R1 => point of invocation.
			07F7	2273	SUBPOP	R1		; Rel. addr of ERROR_LABEL to TOP of STACK
7E	61	32	0801	2274	(VTWL	(R1),-(SP)		; R1 => ERROR_LABEL
51	8E	C0	0804	2275	ADDL	(SP)+,R1		; Branch around.
	11	11	0807	2276	BRB	70\$		:
			0809	2277	60\$: EXTZV	#MF_NDTA_V_NDIC,-		; Extract interrupt code from where
	00	EF	0809	2278		#MF_NDTA_S_NDIC,-		:
	06		080B	2279		UCBSL_MF_NDTA_C(R5),RO		; we stashed it above.
50	0184 C5		080C	2280				:
			0810	2281				:
			0810	2282	SUBPOP	R1		; R1 => point of invocation.
			081A	2283	70\$: ADDL	#2,R1		; Adjust to correct return or error address
51	02	C0	081A	2284	JMP	(R1)		; Return to caller or ERROR_LABEL.
	61	17	081D	2285				:

```
081F 2287 .SBTTL Start READLBLK or READPBLK functions.
081F 2288
081F 2289 ; START_READLBLK and START_READPBLK
081F 2290 ;
081F 2291 ; INPUTS:
081F 2292 ; R3 => TM78 CSR
081F 2293 ; R5 => UCB
081F 2294
081F 2295 START_READLBLK:
081F 2296 START_READPBLK:
081F 2297 BBC #IOSV_REVERSE,- ; If NOT reverse operation, branch
OC 009A 06 E1 0821 2298 UCB$W_FUNC(R5),10$ ; around to handle it.
0825 2299
0825 2300 ; READ REVERSE code
0092 3F 90 0825 2301 MOVB #F_READ_REV!GO_BIT,- ; Save the primary hardware OP CODE in
0827 2302 UCB$B_FEX(R5) ; the UCB.
0093 39 90 082A 2303 MOVB #F_READ_FWD!GO_BIT,- ; And also save the OPPOSITE hardware
082C 2304 UCB$B_CEX(R5) ; OP CODE in the UCB. The OPPOSITE
082F 2305 ; OP CODE is used when we get READ_OPP
082F 2306 ; status back from the controller.
0A 11 082F 2307 BRB 20$ ; Branch around READ FORWARD logic.
0831 2308 10$:
0831 2309
0092 39 90 0831 2310 MOVB #F_READ_FWD!GO_BIT,- ; READ FORWARD code
0833 2311 UCB$B_FEX(R5) ; Save the primary hardware OP CODE in
0093 3F 90 0836 2312 MOVB #F_READ_REV!GO_BIT,- ; the UCB.
0838 2313 UCB$B_CEX(R5) ; And also save the OPPOSITE hardware
083B 2314 ; OP CODE in the UCB. The OPPOSITE
083B 2315 ; OP CODE is used when we get READ_OPP
083B 2316 20$: ; status back from the controller.
0148 30 083B 2317 BSBW READ_OR_WRITECHECK_BLOCK; Call internal subroutine to effect
083E 2318 ; I/O data transfer.
083E 2319
1E 50 E9 083E 2320 BLBC R0,READ_EXIT ; If transfer failed, branch to exit.
0E E1 0841 2321 BBC #IOSV_DATACHECK,- ; If no DATACHECK has been specified
009A C5 0843 2322 UCB$W_FUNC(R5),- ; then we are finished so we branch
18 0846 2323 READ_EXIT ; around to exit.
0847 2324
07 009A 06 E1 0847 2325 BBC #IOSV_REVERSE,- ; If NOT reverse operation, branch
C5 0849 2326 UCB$W_FUNC(R5),30$ ; around to handle it.
084D 2327
084D 2328 ; For READ REVERSE with WRITECHECK,
0092 11 90 084D 2329 MOVB #F_SP_FWD_REC!GO_BIT,- ; we reposition FORWARD one block
C5 084F 2330 UCB$B_FEX(R5) ; so as to be able to do the WRITECHECK
05 11 0852 2331 BRB 40$ ; And we branch around.
0854 2332 30$:
0854 2333
0092 13 90 0854 2334 MOVB #F_SP_REV_REC!GO_BIT,- ; For READ FORWARD with WRITECHECK,
C5 0856 2335 UCB$B_FEX(R5) ; we reposition back one block
0859 2336 40$: ; so as to be able to do the WRITECHECK
05B6 30 0859 2337 BSBW INTERNAL_SPACE ; Execute command in UCB$B_FEX
085C 2338 ; for one tape position.
29 50 E8 085C 2339 BLBS R0,START_WRITECHECK ; If positioning succeeds, branch to
085F 2340 ; execute WRITECHECK. Else fall thru.
085F 2341 READ_EXIT:
0B30 31 085F 2342 BRW FUNCTION_EXIT ; Branch to common exit.
```

```
0862 2344 .SBTTL Start WRITELBLK or WRITEPBLK functions.
0862 2345
0862 2346 ; START_WRITELBLK and START_WRITEPBLK
0862 2347
0862 2348 : INPUTS:
0862 2349 R3 => TM78 CSR
0862 2350 R5 => UCB
0862 2351
0862 2352
0862 2353 START_WRITELBLK:
0862 2354 START_WRITEPBLK:
03D6 30 0862 2355 BSBW WRITE_BLOCK ; Call internal subroutine to effect
0865 2356 ; I/O data transfer.
0865 2357
0878 8F 50 B1 0865 2358 CMPW R0,#SS$_ENDOF TAPE ; ENDOF TAPE is not really an ERROR.
03 13 086A 2359 BEQL 10$ ; therefore branch around if we got it.
086C 2360
16 50 E9 086C 2361 BLBC R0,WRITE_EXIT ; If transfer failed, branch to exit.
086F 2362 10$:
06 64 A5 E0 086F 2363 BBS #UCB$V_ERLOGIP,- ; If we had a recoverable write error,
0871 2364 UCB$W_STS(R5),20$ ; force a DATACHECK operation.
0874 2365
009A 0E E1 0874 2366 BBC #IOSV_DATACHECK,- ; If no DATACHECK has been specified
C5 0876 2367 UCB$W_FUNC(R5),- ; then we are finished so we branch
0B 0879 2368 WRITE_EXIT ; around to exit.
087A 2369
087A 2370 20$:
13 90 087A 2371 MOVB #F_SP_REV_REC!GO_BIT,- ; Else we reposition back one block
0092 C5 087C 2372 UCB$B_FEX(R5) ; so as to be able to do the WRITECHECK
0590 30 087F 2373 BSBW INTERNAL_SPACE ; Execute command in UCB$B_FEX
0882 2374 ; for one tape position.
15 50 E8 0882 2375 BLBS R0,WRITECHECK_FORWARD ; Branch to command WRITECHECK code,
0885 2376 ; if we successfully positioned. Else
0885 2377 ; we fall thru.
0885 2378 WRITE_EXIT:
0B0A 31 0885 2379 BRW FUNCTION_EXIT ; Branch to common exit.
```



```
0888 2381 .SBTTL Start WRITECHECK function.
0888 2382
0888 2383 ; START_WRITECHECK
0888 2384 :
0888 2385 : INPUTS:
0888 2386 : R3 => TM78 CSR
0888 2387 : R5 => UCB
0888 2388 :
0888 2389 :
0888 2390 START_WRITECHECK:
009A 06 E1 0888 2391 BBC #IOSV_REVERSE,- ; If NOT WRITECHECK REVERSE,
088A 2392 UCB$W_FUNC(R5),- ; branch around to handle it.
088D 2393 WRITECHECK_FORWARD
088E 2394
088E 2395 ; WRITECHECK REVERSE.
0092 2F 90 088E 2396 MOVB #F WRITE_CHECK_REV!GO_BIT,- ; Save Primary hardware opcode
0890 2397 UCB$B_FEX(R5) ; in the UCB.
0093 C5 90 0893 2398 MOVB #F WRITE_CHECK_FWD!GO_BIT,- ; Save Secondary hardware OP CODE
0895 2399 UCB$B_CEX(R5) ; in the UCB also.
0A 11 0898 2400 BRB WRITECHECK_COMMON ; And branch around.
089A 2401 WRITECHECK_FORWARD:
089A 2402
0092 29 90 089A 2403 MOVB #F WRITE_CHECK_FWD!GO_BIT,- ; WRITECHECK FORWARD.
089C 2404 UCB$B_FEX(R5) ; Establish primary hardware OP CODE
089F 2405 MOVB #F WRITE_CHECK_REV!GO_BIT,- ; and save it in the UCB.
0093 C5 90 089F 2406 UCB$B_CEX(R5) ; And save OPPOSITE hardware
08A1 2407 WRITECHECK_COMMON: ; in the UCB also.
08A4 2408 BSBW READ OR WRITECHECK_BLOCK ; Effectuate I/O transfer.
43 50 E9 08A7 2409 BLBC R0,WRITECHECK_EXIT ; Branch around if error.
08AA 2410
08AA 2411 ASSUME MBASV_SR_WCKLWR LT 16 ; Assumptions that allow
08AA 2412 ASSUME MBASV_SR_WCKUPR LT 16 ; word compare in following.
B3 08AA 2413 BITW #MBASV_SR_WCKLWR!- ; See if we got a DATACHECK
08AB 2414 #MBASV_SR_WCKUPR!- ; comparison error.
08AB 2415 UCB$L_MFMBA_SR(R5)
0128 C5 0600 8F 08B1 2416 BNEQ 10$ ; NEQ implies DATACHECK error
0C 12 08B3 2417
02 E1 08B3 2418 BBC #MTSV_EOT-16,- ; If NO DATACHECK, see if
46 A5 08B5 2419 UCB$L_DEVDEPEND+2(R5),- ; beyond the EOT on a write.
35 08B7 2420 WRITECHECK_EXIT ; If NOT, then branch around.
50 0878 8F B0 08B8 2421 MOVW #SS$ ENDOFTAPE,R0 ; If SO, return proper status
2E 11 08BD 2422 BRB WRITECHECK_EXIT ; and branch around to exit.
08BF 2423
08BF 2424 10$: ; We had a DATACHECK error.
50 0132 C5 32 08BF 2425 CVTWL UCB$L_MFMBA_BCR+2(R5),R0 ; R0 now contains negative of
08C4 2426 ; # bytes left to transfer.
51 7E A5 3C 08C4 2427 MOVZWL UCB$W_BCNT(R5),R1 ; R1 has length of transfer.
50 51 C0 08C8 2428 ADDL R1,R0 ; R0 = byte # of WORD following erro
50 D7 08CB 2429 DECL R0 ; R0 = relative byte # of 2ND
08CD 2430 ; byte of WORD in error.
29 91 08CD 2431 CMPB #F WRITE_CHECK_FWD!GO_BIT,- ; See if a REVERSE operation.
0092 C5 08CF 2432 UCB$B_FEX(R5) ;
08 12 08D2 2433 BNEQ 20$ ; NEQ implies REVERSE.
08D4 2434
08D4 2435 BBS #MBASV_SR_WCKLWR,- ; For FORWARD operation 1ST
08 0128 C5 08D6 2436 UCB$L_MFMBA_SR(R5),30$ ; byte is LOW order byte.
08 11 08DA 2437 BRB 40$ ; Branch around if 1st byte OK.
```

```
02 0128 0A E1 08DC 2438 20$:          BBC      #MBASV SR WCKUPR -      ; For REVERSE 1ST byte is HIGH.
                                08DC 2439          UCBSL_MFMBA_SR(R5),40$ ; So branch around decrement if
                                08DE 2440          ; it is error free.
                                08E2 2441 30$:          DECL R0      ; R0 = relative byte # of 1ST
                                08E2 2442          ; byte of WORD in error.
                                08E4 2443 40$:          ASHL #16,R0,R0 ; Shift count to hi word of R0.
50 50 10 78 08E4 2445          MOVW #SS$_DATACHECK,R0 ; And indicate DATACHECK error.
50 005C BF B0 08E8 2446          WRITECHECK_EXIT:
                                08ED 2447          BRQ FUNCTION_EXIT ; Branch to common exit.
                                08ED 2448
```

```
08F0 2450 .SBTTL READ TRANSFER ACTION TABLE
08F0 2451
08F0 2452 ; READ_TAT - the following table is used by subroutine READ_OR_WRITECHECK_BLOCK
08F0 2453 ; to determine the actions it should take as the result of the Hardware
08F0 2454 ; Interrupt Code (HIC) that occurs after executing a TM78 read
08F0 2455 ; command. The TAT (Transfer Action Table) contains an entry for
08F0 2456 ; each possible HIC that could be generated by a READ command.
08F0 2457 ;
08F0 2458
08F0 2459 READ_TAT:
08F0 2460
08F0 2461 TAT_ENTRY HIC_DONE,SS$NORMAL,-
08F0 2462 POSITION=YES,COUNT=YES
08FA 2463
08FA 2464 TAT_ENTRY HIC_SHORT_REC,SS$NORMAL,-
08FA 2465 POSITION=YES,COUNT=YES
0904 2466
0904 2467 TAT_ENTRY HIC_LONG_REC,SS$DATAOVERUN,-
0904 2468 POSITION=YES,COUNT=YES
090E 2469
090E 2470 TAT_ENTRY HIC_ERROR,SS$DRVERR,-
090E 2471 POSITION=YES,ERRLOG=YES
0918 2472
0918 2473 TAT_ENTRY HIC_TM,SS$ENDOFFILE,-
0918 2474 DEVDEPEND=MT$M_EOF,-
0918 2475 PREVIM=YES,-
0918 2476 POSITION=YES
0922 2477
0922 2478 TAT_ENTRY HIC_BOT,SS$ENDOFFILE,-
0922 2479 DEVDEPEND=MT$M_BOT,-
0922 2480 PREVIM=YES
092C 2481
092C 2482 TAT_ENTRY HIC_NOT_CAPABLE,SS$OPINCOMPL,-
092C 2483 ERRLOG=YES
0936 2484
0936 2485 TAT_ENTRY HIC_OFF_LINE,SS$MEDOFL
0940 2486
0940 2487 TAT_ENTRY HIC_NON_EX,SS$NONEXDRV
094A 2488
094A 2489 TAT_ENTRY HIC_NOT_AVAIL,SS$DRVERR,-
094A 2490 ERRLOG=YES
0954 2491
0954 2492 TAT_ENTRY HIC_TU_FAULT_A,SS$DRVERR,-
0954 2493 DEVDEPEND=MT$M_LOST,-
0954 2494 ERRLOG=YES
095E 2495
095E 2496 TAT_ENTRY HIC_TM_FAULT_A,SS$CTRLERR,-
095E 2497 DEVDEPEND=MT$M_LOST,-
095E 2498 ERRLOG=YES
0968 2499
0968 2500 TAT_ENTRY HIC_BAD_TAPE,SS$DRVERR,-
0968 2501 DEVDEPEND=MT$M_LOST,-
0968 2502 ERRLOG=YES
0972 2503
0972 2504 TAT_ENTRY HIC_UNREADABLE,SS$DRVERR,-
0972 2505 POSITION=YES,-
0972 2506 ERRLOG=YES
```

TFDRIVER
V04-000

- TM78/TU78 MAGTAPE DRIVER
READ TRANSFER ACTION TABLE

M 8

16-SEP-1984 00:08:15 VAX/VMS Macro V04-00
5-SEP-1984 00:17:37 [DRIVER.SRC]TFDRIVER.MAR;1

Page 57
(1)

TF
VO

097C 2507
097C 2508
097C 2509
097C 2510
097C 2511

TAT_ENTRY

0,SS\$ CTRLERR,-
DEVDEPEND=MTSM_LOST,-
ERRLOG=YES

; END OF TABLE.

```
0986 2513 .SBTTL READ_OR_WRITECHECK_BLOCK
0986 2514
0986 2515
0986 2516 : READ_OR_WRITECHECK_BLOCK - internal subroutine to read or writecheck a
0986 2517 : block.
0986 2518
0986 2519 : INPUTS:
0986 2520 : UCB$B_FEX contains the primary hardware OP CODE to perform.
0986 2521 : UCB$B_CEX contains the OPPOSITE hardware OP CODE to perform in case
0986 2522 : we get READ OPP status.
0986 2523 : IOSV_REVERSE in UCB$W_FUNC is set if the primary OP CODE is a
0986 2524 : reverse operation.
0986 2525 : IOSV_INHRETRY in UCB$W_FUNC is set if we should inhibit RETRY
0986 2526 : attempts.
0986 2527
0986 2528 : RS => UCB
0986 2529
0986 2530 : OUTPUTS:
0986 2531 :
0986 2532
0986 2533
0986 2534 READ_OR_WRITECHECK_BLOCK:
0986 2535 SUBSAVE
0990 2536 REQPCNAN ; Save return point on the SUBSTACK.
0996 2537 MOVL R4,R3 ; Allocate the TM78.
0999 2538 REQSCHAN ; Copy so that R3 => TM78 CSR.
099F 2539 ; Allocate MBA. Returns R4 => MBA CSR.
0092 C5 9B 099F 2540 MOVZBW UCB$B_FEX(R5),- ; Move primary command code and GO_BIT
00B4 C5 09A3 2541 UCB$W_MF_CS1(R5) ; to command template in UCB.
09A6 2542
00B8 C5 B0 09A6 2543 MOVW UCB$W_MF_TMPLTC(R5),- ; Copy template copy of TC register to
00B6 C5 09AA 2544 UCB$W_MF_TC(R5) ; other UCB word.
07 009A C5 E1 09AD 2545 BBC #IOSV_INHRETRY,- ; Branch around setting of SER bit if
8000 8F A8 09AF 2546 UCB$W_FUNC(R5),1$ ; Inhibit RETRY is specified.
00B6 C5 09B3 2547 BISW #MF_TC_M_SER,- ; Set SER bit if INHRETRY specified.
09B7 2548 UCB$W_MF_TC(R5)
098A 2549 1$:
098A 2550 MOVZWL UCB$W_BCNT(R5),-
09BD 2551 MF_BC(R3) ; Copy BYTE COUNT to TM78 register.
09BF 2552 ; NOTICE that we set up the byte count
09BF 2553 ; register before the retry label since
09BF 2554 ; the TM78 retry logic sets an appropriate
09BF 2555 ; value in this register which we would
09BF 2556 ; not not want to reset.
09BF 2557
00C0 C5 D0 09BF 2558 MOVL #MINIMUM_TIMEOUT,- ; Copy TIMEOUT parameter to UCB
09C1 2559 UCB$W_MF_TIMEOUT(R5) ; for safe keeping.
09C4 2560
09C4 2561 READ_WRITECHECK_RETRY:
09C4 2562 MOVL MF_BC(R3),R0 ; Pickup Byte count from device.
09C8 2563 MOVZWL R0,R0 ; Clear extraneous bits.
09CB 2564 BNEQ XXZ ; If not zero, leave as is.
50 00010000 8F D0 09CD 2565 MOVL #16,R0 ; If zero, then treat as if 64K.
09D4 2566 XXZ:
51 7E A5 3C 09D4 2567 MOVZWL UCB$W_BCNT(R5),R1 ; R1 contains buffer size.
51 50 D1 09D8 2568 CMPL R0,R1 ; Insure byte count not > buffer.
04 15 09DB 2569 BLEQ XXZ ; LEQ implies within buffer size.
```

```
14 A3 51 3C 09DD 2570      MOVZWL R1,MF_BC(R3)      ; If buffer smaller, update device reg.
      09E1 2571      XXZZ:
      03C4 30 09E1 2572      BSBW LOAD_MBA_REGISTERS      ; Call to set MBA registers.
      09E4 2573
00B4 C5 3C 09E4 2574      MOVZWL UCBSW_MF_CS1(R5),-      ; Record Hardware command about to
0180 C5 3C 09E8 2575      UCBSL_MF_CMD(R5)      ; become the CURRENT command.
00B6 C5 3C 09EB 2576      MOVZWL UCBSW_MF_TC(R5),-      ;
08 A3 09EF 2577      MF_TC(R3)      ; Copy control info to TM78 register
      09F1 2578      ; NOTICE that this TM78 register is set
      09F1 2579      ; after the RETRY label since there is
      09F1 2580      ; a field in this register that we may
      09F1 2581      ; wish to reset before the retry oper-
      09F1 2582      ; ation. This field is the RECORD
      09F1 2583      ; COUNT field which is set to zero if
      09F1 2584      ; we attempt a READ OPPOSITE.
      09F1 2585
      09F1 2586      DSBINT      ; Stop interrupts.
      09F7 2587      PRIOR_TEST TMRDY      ; Assure that TM78 ready.
42 64 A5 E0 0A00 2588      BBS #UCBSV_POWER,-      ; Make sure we have not just had a
00B4 C5 3C 0A02 2589      UCBSW_STS(R5),3$      ; power failure.
      63 0A05 2590      MOVZWL UCBSW_MF_CS1(R5),-      ;
      0A09 2591      MF_CS1(R3)      ; Copy command to TM78 register.
      0A0A 2592      WFIKPC 4$,UCBSL_MF_TIMEOUT(R5)      ;
      0A16 2593      POST_TEST_TMRDY      ; Assure that TM78 ready.
      0A1F 2594
52 04 A3 D0 0A1F 2595      MOVL MF_IS(R3),R2      ; Copy interrupt status to available reg
      00 EF 0A23 2596      EXTZV #MF_IS_V_DTIC,-      ; Here we extract the interrupt code
      06 0A25 2597      #MF_IS_S_DTIC,-      ; from the interrupt status returned
52 52 0A26 2598      R2,R2      ; by the TM78.
52 09 91 0A28 2599      CMPB #HIC_NOT_RDY,R2      ; Test for MANUAL REWIND or LOADING.
      07 12 0A2B 2600      BNEQ 2$      ; If neither of the above, branch around
      01 A8 0A2D 2601      BISW #UCBSM_MF_REWIND,-      ; If in the middle of manual REWIND,
68 A5 0A2F 2602      UCBSW_DEVSTS(R5)      ; then set appropriate bit.
0905 31 0A31 2603      BRW AWAIT_MANUAL_REWIND      ; Branch to RELEASE CHANNELS, wait for
      0A34 2604      ; rewind or load to finish, and then
      0A34 2605      ; restart the I/O operation.
      0A34 2606
      0D17 30 0A34 2607 2$:      BSBW SAVE_TM78_REGS      ; Save copies of data transfer registers
      0D30 30 0A37 2608      BSBW SAVE_MBA_REGS      ; Save copies of MBA registers
      0A3A 2609      IOF JRK      ; Lower IPL.
      0A40 2610
      05 E1 0A40 2611      BFC #UCBSV_POWER,-      ; Make sure we have not just had a
12 64 A5 0A42 2612      UCBSW_STS(R5),6$      ; power failure.
      0A 11 0A45 2613      JRB 5$      ; Branch around if we had POWERFAIL.
      0A47 2614 3$:      ENBINT
      0A4A 2615 4$:      SETIPL UCBSB_FIPL(R5)
      0A4E 2616      BSBW SAVE_MBA_REGS      ; Save copies of MBA registers
      0422 30 0A51 2617 5$:      BSBW TIMEOUT_POWERFAIL
      012A 31 0A54 2618      BRW READ_BLOCK_EXIT
      C0 EF 0A57 2619 6$:      EXTZV #MF_IS_V_DTIC,-
      06 0A59 2620      #MF_IS_S_DTIC,-
50 0140 C5 0A5A 2621      UCBSL_MF_IS(R5),R0      ; Again we extract the interrupt code
      0A5E 2622      ; from the interrupt status returned
      0A5E 2623      ; by the TM78.
      0A5E 2624      ;
      0A5E 2625      ; The interrupt code is in R0.
      0A5F 2626
```

```
50 01 91 0A5E 2627 CMPB #HIC_DONE,R0 ; See if we received DONE interrupt code.
    12 12 0A61 2628 BNEQ 8$ ; If NOT, branch around.
    0A EF 0A63 2629 EXTZV #MF-IS-V_DTFC,- ; Extract Data Transfer Failure Code
    06 0A65 2630 #MF-IS-S_DTFC,- ; if we had a DONE interrupt code.
51 0140 C5 0A66 2631 UCBSL_MF-IS(R5),R1 ; And put this failure code in R1.
    09 13 0A6A 2632 BEQL 8$ ; EQL implies no failure.
    0100 8F A8 0A6C 2633 BISW #UCBSM_MF_SOFT_ERR,- ; Indicate to DEVICE_ERROR that we had
    68 A5 0A70 2634 UCBSW_DEVSTS(R5) ; a soft error.
    0758 30 0A72 2635 BSBW DEVICE_ERROR ; Do EXTENDED SENSE and log device error
    0A75 2636 8$:
50 12 91 0A75 2637 CMPB #HIC_RETRY,R0 ; See if the HIC must be handled
    0A78 2638 BWEQL 140$ ; outside the Transfer Action Table
50 13 91 0A7D 2639 CMPB #HIC_READ_OPP,R0 ; logic. Those HIC's that are
    0A80 2640 BWEQL 150$ ; handled apart are RETRY, READ_OPP,
50 11 91 0A85 2641 CMPB #HIC_SHORT_REC,R0 ; and SHORT_REC (the latter only in
    0C 12 0A88 2642 BNEQ 10$ ; the case that the last thing done
0093 C5 91 0A8A 2643 CMPB UCBSB_CEX(R5),- ; was a READ_OPP). Therefore if the
00B4 C5 0A8E 2644 UCBSW_MF-CS1(R5) ; HIC = RETRY or READ_OPP, or the
    0A91 2645 BWEQL 160$ ; HIC = SHORT_REC and UCBSB_CEX is
    0A96 2646 ; the current command, we branch.
    0A96 2647
51 FE56 CF 9E 0A96 2648 10$: MOVAB READ_TAT,R1 ; R1 => Read Transfer Action Table.
    61 50 91 0A9B 2650 20$: CMPB R0,TAT_HIC(R1) ; See if current HIC matches TAT entry.
    0A 13 0A9E 2652 BEQL 40$ ; If so, branch out of loop.
    61 95 0AA0 2653 TSTB TAT_HIC(R1) ; See if we ran off end of table.
    06 13 0AA2 2654 BEQL 40$ ; EQL implies End of Table.
51 0A A1 9E 0AA4 2655 MOVAB TAT_LENGTH(R1),R1 ; R1 => next TAT entry.
    F1 11 0AA8 2656 BRB 20$ ; Loop back.
    0AAA 2657 40$:
    00 E1 0AAA 2658 BBC #TAT_V_ERRLOG,- ; Branch around if NO error logging
    03 05 A1 0AAC 2659 TAT_FLAGS(R1),50$ ; for this condition.
    071B 30 0AAF 2660 BSBW DEVICE_ERROR ; Do EXTENDED SENSE and log device error
    0AB2 2661 50$:
50 01 A1 3C 0AB2 2662 MOVZWL TAT_SOFT_STAT(R1),R0 ; Copy status to R0 for user.
    03 A1 A8 0AB6 2663 BISW TAT_DEVDEPEND(R1),- ; Set the appropriate bits in UCB for
    46 A5 0AB9 2664 UCBSL_DEVDEPEND+2(R5) ; the current HIC.
    0ABB 2665
    00 E1 0ABB 2666 BBC #MTSV_BOT-16,- ; If we are NOT at BOT, then we
    08 03 A1 0ABD 2667 TAT_DEVDEPEND(R1),55$ ; branch around.
    00B0 C5 D4 0AC0 2668 CLRL UCBSL_RECORD(R5) ; Else we update position.
    10 AA 0AC4 2669 BICW #<MTSM_LOST>-16,- ; And we are definitely not LOST
    46 A5 0AC6 2670 UCBSL_DEVDEPEND+2(R5) ; anymore.
    0AC8 2671 55$:
    01 E1 0AC8 2672 BBC #TAT_V_POSITION,- ; See if the tape position indicator
    10 05 A1 0ACA 2673 TAT_FLAGS(R1),70$ ; should be updated. Branch if not.
    06 E0 0ACD 2674 BBS #IOSV_REVERSE,- ; A reverse operation implies decrement
06 009A C5 0ACF 2675 UCBSW_FUNC(R5),60$ ; while forward motion means increment.
    00B0 C5 D6 0AD3 2676 INCL UCBSL_RECORD(R5) ; Increment when we read forward.
    04 11 0AD7 2677 BRB 70$ ; Branch around decrement.
    0AD9 2678 60$:
    00B0 C5 D7 0AD9 2679 DECL UCBSL_RECORD(R5) ; Decrement when we read backward.
    0ADD 2680 70$:
    02 E1 0ADD 2681 BBC #TAT_V_COUNT,- ; Branch around if we should NOT return
07 05 A1 0ADF 2682 TAT_FLAGS(R1),80$ ; the byte count of data transferred.
    0150 C5 F0 0AE2 2683 INSV UCBSL_MF_BC(R5),- ; Return the number of bytes
```

```
50 10 10      0AE6 2684      #16,#16,R0      ; transferred in the high word of R0.
              0AE9 2685 80$:      ;
              0AE9 2686      BBC      #TAT V PREVIM,-      ; Branch around if we should NOT
              14 05 A1      0AEB 2687      TAT FLAGS(R1),100$      ; update PREVIM.
              06      E0      0AEE 2688      BBS      #IOSV_REVERSE,-      ; See if operation is reverse.
              09 009A C5      0AF0 2689      UCB$W_FUNC(R5),90$      ; And if so, branch to put negative 2.
              00B0 C5      D0      0AF4 2690      MOVL      UCB$W_RECORD(R5),-      ; Copy current tape position to PREVIM.
              00C8 C5      11      0AF8 2691      UCB$W_MF_PREVIM(R5)      ;
              05      11      0AFB 2692      BRB      100$      ; Branch around reverse motion actions.
              00C8 C5 02      CE      0AFD 2693 90$:      ;
              7D      11      0B02 2694      MNEGL      #2,UCB$W_MF_PREVIM(R5)      ; Indicate unknown previous TAPEMARK.
              11      0B02 2695 100$:      ;
              0B04 2696      BRB      READ_BLOCK_EXIT      ; Branch around to exit.
              0B04 2697      ;
              0B04 2698 140$:      ;
              06C6 30      0B04 2699      BSBW      DEVICE_ERROR      ; Do EXTENDED SENSE and log device error
              01      F0      0B07 2700      INSV      #1,-      ; Set RECORD COUNT to 1 in case it
              02      0B09 2701      #MF_TC_V_RC,-      ; had been reset by a READ OPPOSITE
              06      0B0A 2702      #MF_TC_S_RC,-      ; in a previous RETRY.
              00B6 C5      0B0B 2703      UCB$W_MF_TC(R5)      ;
              0B0E 2704      ;
              0092 C5 9B      0B0E 2705      MOVZBW      UCB$B_FEX(R5),-      ; Copy primary OPCODE to template
              00B4 C5      0B12 2706      UCB$W_MF_CS1(R5)      ; copy of CSR in UCB.
              FEAC 31      0B15 2707      BRW      READ_WRITECHECK_RETRY      ; And branch back to retry.
              0B18 2708      ;
              0B18 2709 150$:      ;
              06B2 30      0B18 2710      BSBW      DEVICE_ERROR      ; Do EXTENDED SENSE and log device error
              0B18 2711      ;
              51 0150 C5      D0      0B18 2712      MOVL      UCB$W_MF_BC(R5),R1      ; R1 = best guess as to record length.
              7E A5 51      B1      0B20 2713      CMPW      R1,UCB$W_BCNT(R5)      ; See if larger than buffer.
              11      1A      0B24 2714      BGTRU      170$      ; If so, branch around.
              0B26 2715 160$:      ; Here we attempt a READ OPPOSITE.
              00FC 8F      AA      0B26 2716      BICW      #MF_TC_M_RC,-      ; Clear the record count prior to READ
              00B6 C5      0B2A 2717      UCB$W_MF_TC(R5)      ; OPPOSITE.
              0093 C5 9B      0B2D 2718      MOVZBW      UCB$B_CEX(R5),-      ; Copy opposite OPCODE to template
              00B4 C5      0B31 2719      UCB$W_MF_CS1(R5)      ; copy of CSR in UCB.
              FE8D 31      0B34 2720      BRW      READ_WRITECHECK_RETRY      ; And branch back to retry.
              0B37 2721      ;
              0B37 2722 170$:      ; Here if record too long for READ OPPOSITE
              0B37 2723      ;
              06      E0      0B37 2724      BBS      #IOSV_REVERSE,-      ; See if we are doing REVERSE operation.
              06 009A C5      0B39 2725      UCB$W_FUNC(R5),173$      ; And if so, branch.
              00B0 C5      D6      0B3D 2726      INCL      UCB$W_RECORD(R5)      ; If FORWARD, increment position count.
              04      11      0B41 2727      BRB      176$      ; And branch around.
              0B43 2728 173$:      ;
              00B0 C5      D7      0B43 2729      DECL      UCB$W_RECORD(R5)      ; Decrement count for REVERSE movement.
              0B47 2730 176$:      ;
              0B47 2731      DSBINT      ; HERE WE GET THE TM78 TO STOP ERROR RETRIES
              0B4D 2732      PRIOR TEST TMRDY      ; Assure that TM78 ready.
              50 54 A5 3C      0B56 2733      MOVZWC      UCB$W_UNIT(R5),R0      ; R0 = unit number of this drive.
              03      3C      0B5A 2734      MOVZWL      #F_NOP!GO_BIT,-      ; Move NDT NOP command to NDT command
              30 A340      0B5C 2735      MF_NDT0(R3)[R0]      ; register.
              0B5F 2736      WFIKPCW 180$,#MINIMUM_TIMEOUT      ; Should be immediate.
              0B69 2737      POST TEST_TMRDY      ; Assure that TM78 ready.
              0B72 2738      IOFORK      ;
              0B78 2739 180$:      ;
              0B78 2740      SETIPL      UCB$B_FIPL(R5)      ; Lower IPL in case of TIMEOUT.
```



```
50 008C 8F 3C 0B7C 2741 MOVZWL #SS$_DRVERR,R0 ; Indicate drive error.
      0B81 2742 READ_BLOCK_EXIT:
      0B81 2743
      0B81 2744 SUBPUSH R0 ; Save R0 on SUBSTACK.
      0B8B 2745
      0B8B 2746 RELCHAN
      0B91 2747
00B0 C5 01 D1 0B91 2748 CMPL #1,UCB$_RECORD(R5) ; See if after first record.
      17 12 0B96 2749 BNEQ 190$ ; If NOT, branch around.
      0B98 2750
      0B98 2751 EX_NDT_CMD CMD=SENSE,- ; If so, Execute a SENSE command
      0B98 2752 TIMEOUT=#MINIMUM_TIMEOUT,-; which should not take much time
      0B98 2753 ERROR_LABEL=190$ ; to determine density.
      0BA7 2754
      50 01 91 0BA7 2755 CMPB #HIC_DONE,R0 ; Was it successful?
      03 12 0BAA 2756 BNEQ 190$ ; If not, branch to ignore.
      0B38 30 0BAC 2757 BSBW RECORD_SENSE_INFO ; Update UCB fields with latest data.
      0BAF 2758 190$:
      0BAF 2759 SUBPOP R0 ; Likewise R0.
      0BB9 2760
      0BB9 2761 SUBRETURN
```

```
.SBTTL WRITE TRANSFER ACTION TABLE
OBC3 2763
OBC3 2764
OBC3 2765 ; WRITE_TAT - the following table is used by subroutine WRITE_BLOCK to
OBC3 2766 ; determine the actions it should take as the result of the Hardware
OBC3 2767 ; Interrupt Code (HIC) that occurs after executing a TM78 write
OBC3 2768 ; command. The TAT (Transfer Action Table) contains an entry for
OBC3 2769 ; each possible HIC that could be generated by a WRITE command.
OBC3 2770 ;
OBC3 2771
OBC3 2772 WRITE_TAT:
OBC3 2773
OBC3 2774 TAT_ENTRY HIC_DONE,SS$NORMAL,-
OBC3 2775 POSITION=YES,COUNT=YES
OBCD 2776
OBCD 2777 TAT_ENTRY HIC_EOT,SS$ENDOFTAPE,-
OBCD 2778 DEVDEPEND=MTSM_EOT,-
OBCD 2779 POSITION=YES,COUNT=YES
OBD7 2780
OBD7 2781 TAT_ENTRY HIC_FPT,SS$WRITLCK,-
OBD7 2782 DEVDEPEND=MTSM_HWL
OBE1 2783
OBE1 2784 TAT_ENTRY HIC_ERROR,SS$DRVERR,-
OBE1 2785 POSITION=YES,ERRLOG=YES
OBE8 2786
OBE8 2787 TAT_ENTRY HIC_BAD_TAPE,SS$DRVERR,-
OBE8 2788 POSITION=YES,ERRLOG=YES,-
OBE8 2789 SUBCODE_RUT=BAD_TAPE_SUBROUT
OBF5 2790
OBF5 2791 TAT_ENTRY HIC_EOT_ERROR,SS$DRVERR,-
OBF5 2792 DEVDEPEND=MTSM_EOT,-
OBF5 2793 POSITION=YES,ERRLOG=YES
OBF8 2794
OBF8 2795 TAT_ENTRY HIC_NON_EX,SS$NONEXDRV,-
OBF8 2796 ERRLOG=YES
OC09 2797
OC09 2798 TAT_ENTRY HIC_NOT_AVAIL,SS$DRVERR,-
OC09 2799 ERRLOG=YES
OC13 2800
OC13 2801 TAT_ENTRY HIC_OFF_LINE,SS$MEDOFL
OC1D 2802
OC1D 2803 TAT_ENTRY HIC_TU_FAULT_A,SS$DRVERR,-
OC1D 2804 DEVDEPEND=MTSM_LOST,-
OC1D 2805 ERRLOG=YES
OC27 2806
OC27 2807 TAT_ENTRY HIC_TM_FAULT_A,SS$CTRLERR,-
OC27 2808 DEVDEPEND=MTSM_LOST,-
OC27 2809 ERRLOG=YES
OC31 2810
OC31 2811 TAT_ENTRY O,SS$CTRLERR,- ; END OF TABLE.
OC31 2812 DEVDEPEND=MTSM_LOST,-
OC31 2813 ERRLOG=YES
```

```
0C3B 2815 .SBTTL WRITE_BLOCK
0C3B 2816
0C3B 2817 ; WRITE_BLOCK - internal subroutine to write a block.
0C3B 2818
0C3B 2819 : INPUTS:
0C3B 2820 : IOSV_INHRETRY in UCBSW_FUNC is set if we should inhibit RETRY
0C3B 2821 :
0C3B 2822 : R5 => UCB
0C3B 2823 :
0C3B 2824 : OUTPUTS:
0C3B 2825 :
0C3B 2826 :
0C3B 2827 :
0C3B 2828 WRITE_BLOCK:
0C3B 2829 SUBSAVE ; Save return point on the SUBSTACK.
0C45 2830 REQPCN ; Allocate the TM78.
53 54 D0 0C4B 2831 MOVL R4,R3 ; Copy so that R3 => TM78 CSR.
0C4E 2832 REQSCHAN ; Allocate MBA. Returns R4 => MBA CSR.
50 31 9A 0C54 2833 MOVZBL #F_WRITE_PE!GO_BIT,R0 ; Load write 1600 BPI command to R0.
00BA C5 95 0C57 2834 TSTB UCBSB_MF_DENSITY(R5) ; Zero implies 1600 BPI on tape.
0C5B 2835 ASSUME MF$K_DENSITY_1600 EQ 0
50 03 13 0C5B 2836 BEQL 10$ ; Branch if it is indeed 1600.
50 33 9A 0C5D 2837 MOVZBL #F_WRITE_GCR!GO_BIT,R0 ; Else load write 6250 BPI command in R0
00B4 C5 50 B0 0C60 2838 10$: MOVW R0,UCBSW_MF_CS1(R5) ; Remember command in UCB.
0C65 2840
00B8 C5 B0 0C65 2841 MOVW UCBSW_MF_TMPLTC(R5),- ; Copy template copy of TC register to
00B6 C5 0C69 2842 UCBSW_MF_TC(R5) ; other UCB word.
07 009A C5 0F E1 0C6C 2843 BBC #IOSV_INHRETRY,- ; Branch around setting of SER bit if
8000 8F A8 0C6E 2844 UCBSW_FUNC(R5),20$ ; Inhibit RETRY is specified.
00B6 C5 0C72 2845 BISW #MF_TC_M_SER,- ; Set SER bit if INHRETRY specified.
0C76 2846 UCBSW_MF_TC(R5)
0C79 2847 20$: MOVL #MINIMUM_TIMEOUT,- ; Copy TIMEOUT parameter to UCB
00C0 C5 D0 0C79 2848 UCBSL_MF_TIMEOUT(R5) ; for safe keeping.
7E A5 3C 0C7E 2849 MOVZWL UCBSW_BCNT(R5),-
14 A3 0C81 2851 MF_BCTR3) ; Copy BYTE COUNT to TM78 register.
0C83 2852
00B6 C5 3C 0C83 2853 MOVZWL UCBSW_MF_TC(R5),-
08 A3 0C87 2854 MF_TCTR3) ; Copy control info to TM78 register
0C89 2855 WRITE_RETRY:
011C 30 0C89 2856 BSBW LOAD MBA_REGISTERS ; Set MBA registers.
00B4 C5 3C 0C8C 2857 MOVZWL UCBSW_MF_CS1(R5),- ; Record Hardware command about to
0180 C5 0C90 2858 UCBSL_MF_CMD(R5) ; become the CURRENT command.
AA 0C93 2859 BICW #<MTSM_BOT!- ; Clear beginning of tape and,
0C94 2860 MTSM_EOF!- ; End of File and
0C94 2861 MTSM_EOT!- ; End of Tape and
0C94 2862 MTSM_HWL>2-16,- ; Hardware Write Lock
46 A5 0F 0C94 2863 UCBSL_DEVDEPEND+2(R5)
0C97 2864 DSBINT ; Stop interrupts.
0C9D 2865 PRIOR_TEST_TMRDY ; Assure that TM78 ready.
42 64 A5 E0 0CA6 2866 BBS #UCBSV_POWER,- ; Make sure we have not just had a
00B4 C5 3C 0CA8 2867 UCBSW_STS(R5),40$ ; power failure.
63 0CAF 2868 MOVZWL UCBSW_MF_CS1(R5),-
0CB0 2869 MF_CST(R3) ; Copy command to TM78 register.
0CBC 2870 WFIKPCH 50$,UCBSL_MF_TIMEOUT(R5)
0CBC 2871 POST_TEST_TMRDY ; Assure that TM78 ready.
```

```
52 04 A3 D0 OCC5 2872      MOVL MF_IS(R3),R2      ; Interrupt status to available register
      00 EF OCC9 2873      EXTZV #MF_IS_V_DTIC,-      ; Here we extract the interrupt code
      06      OCCB 2874      #MF_IS_S_DTIC,-      ; from the interrupt status returned
52 52      OCCC 2875      R2,R2      ; by the TM78.
52 09 91 OCCE 2876      CMPB #HIC_NOT_RDY,R2      ; Test for manual REWIND or LOADING.
      07 12 OCD1 2877      BNEQ 30$      ; If neither of the above, branch around
      01 A8 OCD3 2878      BISW #UCBSM MF_REWIND,-      ; If in the middle of manual REWIND,
68 A5      OCD5 2879      UCBSW_DEVSTS(R5)      ; then set appropriate bit.
      06 31 OCD7 2880      BRW AWAIT_MANUAL_REWIND      ; Branch to RELEASE CHANNELS, wait for
      06 31 OCDA 2881      ; rewind or load to finish, and then
      06 31 OCDA 2882      ; restart the I/O operation.
      06 31 OCDA 2883
      06 31 OCDA 2884
      0A 30 OCDA 2885 30$: BSBW SAVE_TM78_REGS      ; Save copies of data transfer registers
      0A 30 OCDD 2886      BSBW SAVE_MBA_REGS      ; Save copies of MBA registers
      0A 30 OCE0 2887      IOFORK      ; Lower IPL.
      0A 30 OCE6 2888
      05 E1 OCE6 2889      BBC #UCBSV_POWER,-      ; Make sure we have not just had a
12 64 A5      OCE8 2890      UCBSW_STS(R5),60$      ; power failure.
      0A 11 OCEB 2891      BRB 55$      ; Branch around if POWERFAIL.
      0A 11 OCED 2892 40$: ENBINT
      0A 11 OCFC 2893 50$: SETIPL UCBSB_FIPL(R5)
      0A 30 OCF4 2894      BSBW SAVE_MBA_REGS      ; Save copies of MBA registers
      01 30 OCF7 2895 55$: BSBW TIMEOUT_POWERFAIL
      07 31 OCFA 2896      BRW WRITE_BLOCK_EXIT
      00 EF OCFD 2897 60$: EXTZV #MF_IS_V_DTIC,-      ; Again we extract the interrupt code
      06      OCFF 2898      #MF_IS_S_DTIC,-      ; from the interrupt status returned
50 0140 C5      OD00 2899      UCBSL_MF_IS(R5),R0      ; by the TM78.
      06      OD04 2900
      06      OD04 2901
      06      OD04 2902 ; The interrupt code is in R0.
      06      OD04 2903
      06      OD04 2904
      06      OD04 2905
50 01 91 OD04 2905      CMPB #HIC_DONE,R0      ; See if we received DONE interrupt code.
      12 12 OD07 2906      BNEQ 65$      ; If NOT, branch around.
      0A EF OD09 2907      EXTZV #MF_IS_V_DTFC,-      ; Extract Data Transfer Failure Code
      06      OD0B 2908      #MF_IS_S_DTFC,-      ; if we had a DONE interrupt code.
51 0140 C5      OD0C 2909      UCBSL_MF_IS(R5),R1      ; And put this failure code in R1.
      09 13 OD10 2910      BEQL 65$      ; EQL implies no failure.
      01 8F A8 OD12 2911      BISW #UCBSM MF_SOFT_ERR,-      ; Indicate to DEVICE_ERROR that we had
68 A5      OD16 2912      UCBSW_DEVSTS(R5)      ; a soft error.
      04 30 OD18 2913      BSBW DEVICE_ERROR      ; Do EXTENDED SENSE and log device error
      04 30 OD1B 2914 65$:
50 12 91 OD1B 2915      CMPB #HIC_RETRY,R0      ; Should we attempt a retry?
      06 12 OD1E 2916      BNEQ 70$      ; If so, branch around.
      04 AA 30 OD20 2917      BSBW DEVICE_ERROR      ; Do EXTENDED SENSE and log device error
      FF 31 OD23 2918
      63 31 OD23 2919      BRW WRITE_RETRY      ; Branch back to try again.
      63 31 OU26 2920
      63 31 OD26 2921
      63 31 OD26 2922 ; Here we use the WRITE_TAT (Write Transfer Action Table) to determine the
      63 31 OD26 2923 ; actions to take as a result of the WRITE COMMAND we have just executed.
      63 31 OD26 2924
      63 31 OD26 2925
      63 31 OD26 2926 70$:
51 FE99 CF 9E OD26 2927      MOVAB WRITE_TAT,R1      ; R1 => WRITE Transfer Action Table.
      63 31 OD2B 2928 80$:
```

```

        61  50  91  0D2B  2929      CMPB   RO,TAT_HIC(R1)      ; See if this TAT entry corresponds to
                                0D2E  2930                      ; the current Hardware Interrupt Code.
                                0A  13  0D2E  2931      BEQL   100$      ; If so, branch out of loop.
                                61  95  0D30  2932      TSTB   TAT_HIC(R1) ; Test for end of table.
                                06  13  0D32  2933      BEQL   100$      ; EQL implies End of Table.
51  0A  A1  9E  0D34  2934      MOVAB  TAT_LENGTH(R1),R1      ; R1 => next TAT entry.
        F1  11  0D38  2935      BRB    80$      ; And branch back to loop start.
                                0D3A  2936 100$:      ; If here, R1 => correct TAT entry.
                                00  E1  0D3A  2937      BBC    #TAT_V_ERRLOG,- ; Branch around if NO error logging
03  05  A1  0D3C  2938      TAT_FLAGS(R1),110$      ; for this condition.
        048B  30  0D3F  2939      BSBW  DEVICE_ERROR      ; Do EXTENDED SENSE and log device error
                                0D42  2940 110$:
50  01  A1  3C  0D42  2941      MOVZWL TAT_SOFT_STAT(R1),R0      ; Copy status to R0 for user.
        03  A1  A8  0D46  2942      BISW  TAT_DEVDEPEND(R1),- ; Set the appropriate bits in UCB for
        46  A5  0D49  2943      UCB$L_DEVDEPEND+2(R5) ; the current HIC.
                                01  E1  0D4B  2944      BBC    #TAT_V_POSITION,- ; See if the tape position indicator
04  05  A1  0D4D  2945      TAT_FLAGS(R1),120$      ; should be updated. Branch if not.
        00B0  C5  D6  0D50  2946      INCL  UCB$L_RECORD(R5) ; Increment since we always write forward.
                                0D54  2947 120$:
                                02  E1  0D54  2948      BBC    #TAT_V_COUNT,- ; Branch around if we should NOT return
07  05  A1  0D56  2949      TAT_FLAGS(R1),130$      ; the byte count of data transferred.
50  10  10  F0  0D59  2950      INSV  UCB$L_MF_BC(R5),- ; Return the number of bytes
                                0D5D  2951      #16,#16,R0 ; transferred in the high word of R0.
                                0D60  2952 130$:
                                06  A1  D5  0D60  2953      TSTL  TAT_SUBCODE_RUT(R1) ; Is the subcode nonzero
                                0A  13  0D63  2954      BEQL  140$      ; If EQL then zero
                                0D65  2955
                                0D65  2956 ; The following code is here to relocate the address stored in TAT_SUBCODE_RUT
                                0D65  2957 ; to be the real address of the routine to call.
                                0D65  2958
        F297  CF  9F  0D65  2959      PUSHAB TF$DDT
6E  06  A1  C0  0D69  2960      ADDL  TAT_SUBCODE_RUT(R1),(SP)
        9E  16  0D6D  2961      JSB   @ (SP)+
                                0D6F  2962
                                0D6F  2963 140$:
                                0D6F  2964 WRITE_BLOCK_EXIT:
                                0D6F  2965
                                0D6F  2966      SUBPUSH R0      ; Save R0 on SUBSTACK.
                                0D79  2967
                                0D79  2968      RELCHAN      ; Release all channels.
                                0D7F  2969
                                0D7F  2970      SUBPOP  R0      ; Likewise R0.
                                0D89  2971
                                0D89  2972      SUBRETURN      ; Return to caller.
                                0D93  2973
                                0D93  2974 BAD_TAPE_SUBROUT:
51  06  0A  EF  0D93  2975      EXTZV  #MF_IS_V_DTFC,#MF_IS_S_DTFC,- ; Get the subcode
        0140  C5  0D96  2976      UCB$L_MF_IS(R5),RT
        51  0E  91  0D9A  2977      CMPB   #HFC_EOT,R1      ; IF NEQ then not EOT subcode
                                08  12  0D9D  2978      BNEQ  10$      ; No need to handle it
44  A5  00040000 8F  C8  0D9F  2979      BISL  #MTSM_EOT,UCB$L_DEVDEPEND(R5) ; Else set EOT in UCB
                                05  05  0DA7  2980 10$:      RSB
                                0DA8  2981
```

```
ODA8 2983      .SBTTL LOAD_MBA_REGISTERS
ODA8 2984
ODA8 2985
ODA8 2986 : LOAD_MBA_REGISTERS
ODA8 2987 :
ODA8 2988 : INPUTS:
ODA8 2989 :   R3 => TM78_CSR
ODA8 2990 :   UCBSW_MF_CS1(R5) = HARDWARE_OPCODE FOR DATA TRANSFER
ODA8 2991 :   UCBSW_BOFF(R5) = BYTE_OFFSET IN PAGE OF TRANSFER
ODA8 2992 :   MF_BC(R3) = BYTE COUNT
ODA8 2993 :   R4=> MBA_CSR
ODA8 2994 :       it is assumed that both channels are owned.
ODA8 2995 :
ODA8 2996 : OUTPUTS:
ODA8 2997 :   MBA_MAP_REGISTERS AND MBASL_VAR ARE LOADED.
ODA8 2998 : Note this internal subroutine does NOT do a SUBSAVE upon entry and a
ODA8 2999 : SUBRETURN at exit since it is assured that it cannot lose control
ODA8 3000 : during its execution. It can therefore use the normal stack to
ODA8 3001 : save its return point and temporary variables.
ODA8 3002 :
ODA8 3003 :
ODA8 3004 : LOAD_MBA_REGISTERS:
ODA8 3005 :
08 A4 00 D2 ODA8 3006 : MCOML #0,MBASL_SR(R4) ; Clear MASSBUS adapter errors.
ODAC 3007 : LOADMBA ; Load MBA map registers.
ODB2 3008 :
ODB2 3009 :
ODB2 3010 : The following section of code was put here to correct a timing glitch that
ODB2 3011 : introduces errors on systems where the TM78 is connected to the MBA by
ODB2 3012 : a MASSBUS cable less than 100 feet in length. The problem occurs in
ODB2 3013 : writing odd length records. Apparently the MBA shuts down before
ODB2 3014 : transferring the last byte. The work around is to load the MBA byte
ODB2 3015 : count register with a count one greater than the desired transfer
ODB2 3016 : length. The TM78 byte count register still contains the correct count.
ODB2 3017 : If this incrementation of the transfer length happens to overflow the
ODB2 3018 : apparent transfer to a new page, then this must be handled by providing
ODB2 3019 : a valid MBA map register value in the currently last position (i.e. the
ODB2 3020 : one with the invalid entry) and by moving the invalid value down one
ODB2 3021 : slot. Since the contents of this new last entry is not pertinent, (i.e.
ODB2 3022 : it only needs to be valid so as to not cause an invalid access) we
ODB2 3023 : merely replicate the current last valid value into the new last position.
ODB2 3024 : All of this is, needless to say, extremely ugly!!!
ODB2 3025 :
ODB2 3026 :
51 14 A3 D0 ODB2 3027 : MOVL MF_BC(R3),R1 ; Get byte count from TM78 register.
51 51 3C ODB6 3028 : MOVZWL R1,R1 ; Erase high word of register.
41 51 E9 ODB9 3029 : BLBC R1,20$ ; If EVEN byte count branch around
ODBC 3030 : ; timing glitch workaround.
ODBC 3031 : CMPB #F_WRITE_PE!GO_BIT,- ; If ODD byte count, see if a WRITE
00B4 C5 ODBE 3032 : UCBSW_MF_CS1(R5) ; operation at 1600 BPI.
07 13 ODC1 3033 : BEQL 10$ ; If yes, WRITE branch.
33 91 ODC3 3034 : CMPB #F_WRITE_GCR!GO_BIT,- ; Still ODD byte count see if WRITE
00B4 C5 ODC5 3035 : UCBSW_MF_CS1(R5) ; operation at 6250 BPI.
33 12 ODC8 3036 : BNEQ 20$ ; If NOT write of any kind branch.
ODCA 3037 10$: ;
ODCA 3038 : ; If here we have an ODD byte count on
ODCA 3039 : ; a WRITE operation
```

50	7C	A5	3C	ODCA	3040	MOVZWL	UCB\$W_BOFF(R5),R0	:	We must determine if we will overflow
				ODCE	3041			:	onto a new page by bumping count.
	50	51	C0	ODCE	3042	ADDL	R1,R0	:	R0 contains MBA virtual address of
				ODD1	3043			:	last byte of transfer including
				ODD1	3044			:	increment.
				ODD1	3045			:	
		51	D6	ODD1	3046	INCL	R1	:	Increment count to next higher even.
	10	A4	51	CE	ODD3	MNEGL	R1,MBASL_BCR(R4)	:	Move incremented negated count to MBA.
				ODD7	3048			:	
00	50	09	00	ED	ODD7	CMPZV	#0,#9,R0,#0	:	If overflowed, then low order is zero.
			33	12	ODDC	BNEQ	40\$	:	If not zero, then ok, so branch around
50	50	F7	8F	78	ODDE	ASHL	#-9,R0,R0	:	Get # valid MBA map registers used.
51	0800	C440		D0	ODE3	MOVL	MBASL_MAP(R4)[R0],R1	:	Get invalid map register value.
0804	C440	51		D0	ODE9	MOVL	R1,MBASL_MAP+4(R4)[R0]	:	And move it up one position.
51	07FC	C440		D0	ODEF	MOVL	MBASL_MAP-4(R4)[R0],R1	:	Now get last valid entry.
0800	C440	51		D0	ODF5	MOVL	R1,MBASL_MAP(R4)[R0]	:	And also slide it up one slot.
		14	11		ODFB	BRB	40\$	:	And branch around rest since we
					ODFD			:	know that we have no REVERSE.
					ODFD			:	
					ODFD			:	
					ODFD			:	
		3F	91		ODFD	CMPB	#F_READ_REV!GO_BIT,-	:	
00B4	C5				ODFF		UCB\$W_MF_CS1(R5)	:	Is it a REVERSE read?
	07	13			OE02	BEQL	30\$	:	If so, branch.
	2F	91			OE04	CMPB	#F_WRITE_CHECK_REV!GO_BIT,-	:	
00B4	C5				OE06		UCB\$W_MF_CS1(R5)	:	Is it a REVERSE writecheck?
	06	12			OE09	BNEQ	40\$	:	If NOT, then skip around
					OE0B			:	since MBASL_VAR is ok.
					OE0B			:	
		51	D7		OE0B	DECL	R1	:	Decrement transfer byte count.
0C	A4	51	C0		OE0D	ADDL	R1,MBASL_VAR(R4)	:	Calculate ending address of buffer
					OE11			:	for REVERSE operation.
					OE11			:	
			05		OE11	RSB		:	Return to caller.
					3073			:	

```
OE12 3075      .SBTTL  INTERNAL_SPACE
OE12 3076
OE12 3077      : INTERNAL_SPACE - subroutine to space the tape one position.
OE12 3078
OE12 3079      : INPUTS:
OE12 3080
OE12 3081      :       UCB$B_FEX(R5) - contains hardware command; either F_SP_FWD_REC!GO_BIT or
OE12 3082      :       F_SP_REV_REC!GO_BIT
OE12 3083
OE12 3084      : OUTPUTS:
OE12 3085
OE12 3086      :       R0 = success or failure code.
OE12 3087      :       UCB$L_RECORD is updated if the tape is moved.
OE12 3088
OE12 3089
OE12 3090      : INTERNAL_SPACE:
OE12 3091
OE12 3092      : SUBSAVE
50 0092 C5 9A OE1C 3093      : MOVZBL  UCB$B_FEX(R5),R0      ; Save return point on SUBSTACK.
OE21 3094      :                               ; Copy hardware command to R0.
53 24 A5 D0 OE21 3095      : MOVL     UCB$L_CRB(R5),R3      ; R3=>TM78-CRB.
OE25 3096      : ASSUME   IDB$L_CSR  EQ 0
53 2C B3 D0 OE25 3097      : MOVL     @CRB$[INTD+VEC$L_IDB(R3),R3  ; R3=>TM78-CSR.
OE29 3098
OE29 3099      : EX_NDT_CMD      CMD=R0,-      ; Execute space command
OE29 3100      : REPEAT=#1,-      ; to space one position
OE29 3101      : TIMEOUT=#MINIMUM_TIMEOUT,-; which shouldn't take long.
OE29 3102      : ERROR_LABEL=50$
OE3D 3103
OE3D 3104      :
OE3D 3105      : HARDWARE STATUS RETURNED IN R0.
OE3D 3106
OE3D 3107
50 01 91 OE3D 3108      : CMPB     #HIC_DONE,R0      ; Done successfully?
16 12 OE40 3109      : BNEQ     30$      ; If not, branch around.
13 91 OE42 3110      : CMPB     #F_SP_REV_REC!GO_BIT,-  ; If yes, see if we moved forward or
0092 C5 OE44 3111      : UCB$B_FEX(R5)      ; backward.
06 13 OE47 3112      : BEQL     10$      ; If backward, branch.
00B0 C5 D6 OE49 3113      : INCL     UCB$L_RECORD(R5)      ; If forward, increment position count.
04 11 OE4D 3114      : BRB      20$      ; and branch around.
00B0 C5 D7 OE4F 3115 10$: : DECL     UCB$L_RECORD(R5)      ; If backward, decrement.
50 01 3C OE53 3116 20$: : MOVZWL  S^#SS$_NORMAL,R0      ; Indicate successful spacing operation.
14 11 OE56 3117      : BRB      50$      ; And branch to exit.
OE58 3118
OE58 3119 30$: :
0372 30 OE58 3120      : BSBW     DEVICE_ERROR      ; Here if some kind of error.
OE58 3121      : Do EXTENDED SENSE and log device error
50 18 91 OE5B 3122      : CMPB     #HIC_TM_FAULT_A,R0    ; See if it was a controller failure.
07 12 OE5E 3123      : BNEQ     40$      ; If not, branch around.
50 0054 8F 3C OE60 3124      : MOVZWL   #SS$_CTRLERR,R0      ; Else indicate controller problem.
05 11 OE65 3125      : BRB      50$      ; And branch to common exit.
OE67 3126 40$: :
50 008C 8F 3C OE67 3127      : MOVZWL   #SS$_DRVERR,R0      ; Indicate drive error.
OE6C 3128 50$: : SUBRETURN
```



```
OE76 3130 .SBTTL TIMEOUT_POWERFAIL
OE76 3131
OE76 3132 : TIMEOUT_POWERFAIL - we call here if we have experienced either a TIMEOUT
OE76 3133 : of a POWERFAILURE.
OE76 3134
OE76 3135 INPUTS:
OE76 3136
OE76 3137 R3 => TM78 CSR
OE76 3138 R5 => UCB
OE76 3139
OE76 3140 OUTPUTS:
OE76 3141
OE76 3142 : In the case of TIMEOUT, various things are recorded in the UCB:
OE76 3143
OE76 3144
OE76 3145 : In the case of POWERFAIL,
OE76 3146
OE76 3147
OE76 3148
OE76 3149
OE76 3150
OE76 3151
OE76 3152 TIMEOUT_POWERFAIL:
OE76 3153 REPOSITION:
01 01 E1 OE76 3154 BBC #UCBSV MF POWER,- : Do NOT enter this routine RECURSIVELY.
01 68 A5 OE78 3155 UCBSW_DEVSTS(R5),10$ : Branch around only if bit clear.
05 OE7B 3156 RSB : Return immediately if RECURSIVE call.
OE7C 3157 10$:
02 A8 OE7C 3158 BISW #UCBSM MF POWER,- : Set bit to prevent RECURSIVE entry.
68 A5 OE7E 3159 UCBSW_DEVSTS(R5)
OE80 3160 SUBSAVE : Pop return off stack and onto SUBSTACK
OE8A 3161 RELCHAN : Release any channels.
01 AA OE90 3162 BICW #UCBSM MF REWIND,-
68 A5 OE92 3163 UCBSW_DEVSTS(R5) : Clear rewind in progress.
10 A8 OE94 3164 BISW #<MTSM LOST@-16>,-
46 A5 OE96 3165 UCBSL_DEVDEPEND+2(R5) : Set position lost.
05 E4 OE98 3166 BBSC #UCBSV POWER,-
68 64 A5 OE9A 3167 UCBSW_STS(R5),POWERFAIL : Branch and clear if powerfail.
03 E0 OE9D 3168 BBS #UCBSV MF REPOS,- : Treat REPOSITIONING request as if
68 A5 OE9F 3169 UCBSW_DEVSTS(R5),- : it were a POWERFAIL.
63 OEA1 3170 POWERFAIL
OEA2 3171 TIMEOUT:
08A9 30 OEA2 3172 BSBW SAVE TM78 REGS : Save TM78 regs for ERROR LOG
00000000'GF 16 OEA5 3173 JSB G^ER[ $DEVICTMO : Log device time out.
OEA8 3174
54 24 A5 D0 OEA8 3175 MOVL UCBSL_CRB(R5),R4 : R4 => TM78 CRB.
54 20 A4 D0 OEAF 3176 MOVL CRBSL_LINK(R4),R4 : R4 => MBA CRB.
54 2C A4 D0 OEB3 3177 MOVL CRBSL_INTD+VEC$[L_IDB(R4),R4 : R4 => MBA IDB
04 A4 55 D1 OEB7 3178 CMPL R5,IDBSL_OWNER(R4) : Are we owners of the MBA.
25 12 OEBB 3179 BNEQ 20$ : If not, branch around.
OEBD 3180
54 64 D0 OEBD 3181 MOVL IDBSL_CSR(R4),R4 : R4 => MBA CSR.
OEC0 3182 DSBINT : We want to abort the operation.
D0 OEC6 3183 MOVL #MBASM_CR_ABORT!-
OEC 3184 MBASM_CR-IE,- : Set abort and interrupt enable
04 A4 06 OEC7 3185 MBASL_CR(R4) : In MBA control register.
OFLA 3186 WFIKPC 10$,#T5 : Wait for abort and keep channel.
```

```
0ED4 3187 IOFORK
0EDA 3188 10$:
01 D0 0EDA 3189 MOVL #MBASH CR INIT,-
04 A4 0EDC 3190 MBASH CR(R4) ; Initialize entire MBA.
04 04 D0 0EDE 3191 MOVL #MBASH CR IE,-
04 A4 0EE0 3192 MBASH CR(R4) ; And enable MBA interrupts.
0EE2 3193 20$: SETIPL UCBSB_FIPL(R5) ; Lower IPL to device level.
0EE6 3194
0EE6 3195 TIMEOUT_RETURN:
0EE6 3196 RELCHAN ; In case we own the channel(s).
02 AA 0EEC 3197 BICW #UCBSM MF POWER,- ; Clear bit to prevent RECURSIVE entry.
68 A5 0EEE 3198 UCBSW_DEVSTS(R5)
0800 8F AA 0EF0 3199 BICW #UCBSM VALID,-
64 A5 0EF4 3200 UCBSW_STS(R5) ; Set volume - SOFTWARE INVALID.
50 022C 8F 3C 0EF6 3201 MOVZWL #SSS_TIMEOUT,R0 ; Indicate final status.
0EF8 3202 SUBRETURN ; Return to caller with TIMEOUT status.
0F05 3203
0F05 3204 POWERFAIL:
0F05 3205 RELCHAN ; In case we own the channel(s).
08 AA 0F0B 3206 BICW #UCBSM MF REPOS,- ; Clear bit in case also at POWERFAIL.
68 A5 0F0D 3207 UCBSW_DEVSTS(R5)
08 E1 0F0F 3208 BBC #UCBSV VALID,-
64 A5 0F11 3209 UCBSW_STS(R5),-
D2 0F13 3210 TIMEOUT_RETURN ; If invalid, treat as TIMEOUT.
0F14 3211
0F14 3212 REWIND WAIT=YES,ERROR_LABEL=10$; Here we try to rewind the drive
0F1E 3213 ; if we still have vacuum in the
0F1E 3214 ; columns. If it works we branch to
50 01 91 0F1E 3215 CMPB #HIC_DONE,R0 ; 80$. The status is returned in R0.
0F21 3216 BWEQL 80$ ; If it doesn't work for any reason we
0F26 3217 ; wind up at 10$.
0F26 3218
0F26 3219 ;
0F26 3220 ; Here we will wait for the drive to be rewound manually.
0F26 3221 ;
0F26 3222
0093 C5 94 0F26 3223 10$: CLRB UCBSB_CEX(R5) ; Clear byte we use for message time counter
53 24 A5 D0 0F2A 3224 20$: MOVL UCBSL_CRB(R5),R3 ; R3 => TM78 CRB.
0F2E 3225 ASSUME IDBSL_CSR EQ 0
53 2C B3 D0 0F2E 3226 MOVL @CRBSL_INTD+VECSL_IDB(R3),R3 ; R3 => TM78 CSR
0F32 3227
0F32 3228 DSBINT
0F38 3229 WFIKPC 30$,#2 ; Waste some time.
0F42 3230 IOFORK
0F48 3231 30$: SETIPL UCBSB_FIPL(R5) ; Lower IPL to fork level.
0F4C 3232 REQPC 30$,#2 ; Sequentialize access to TM78.
0F52 3233 BWS #UCBSV MF CNCLP,- ; If CANCEL pending set while we were
0F52 3234 UCBSW_DEVSTS(R5),190$ ; resource waiting, branch around to end.
0F5A 3235
0F5A 3236 DSBINT
0F60 3237 PRIOR_TEST TMRDY ; Here we will get SENSE info.
05 E4 0F69 3238 BBSC #UCBSV POWER,- ; Assure that TM78 ready.
41 64 A5 0F6B 3239 UCBSW_STS(R5),40$ ; Branch if another POWERFAIL.
50 54 A5 3C 0F6E 3240 MOVZWL UCBSW_UNIT(R5),R0 ; R0 = unit number of this drive.
09 9A 0F72 3241 MOVZBL #F_SENSE!GO_BIT,- ; Record Hardware command about to
0180 C5 0F74 3242 UCBSL MF CMD(R5) ; become the CURRENT command.
09 9A 0F77 3243 MOVZBL #F_SENSE!GO_BIT,-
```

```
30 A340      0F79 3244      MF_NDT0(R3)[R0]      ; Set command in device.
              0F7C 3245      WFIKPC 50$,#MINIMUM_TIMEOUT ; Wait for interrupt.
              0F86 3246      POST_TEST TMRDY          ; Assure that TM78 ready.
0168 C5      0F8F 3247      MOVL UCB$M_MF_NDTA(R5),-    ; Copy NDT attention data for this
0184 C5      0F93 3248      MOVL UCB$M_MF_NDTA_C(R5),-  ; interrupt to stable place.
1C A3      0F96 3249      MOVL MF_DSTR3,-
0158 C5      0F99 3250      MOVL UCB$M_MF_DS(R5)        ; Save pertinent register from SENSE.
              0F9C 3251      IOFORK
              0FA2 3252      BBSC #UCB$V_POWER,-
0B 64 A5      0FA4 3253      UCB$W_STS(R5),45$          ; Branch if another POWERFAIL.
              0FA7 3254      RELCHAN
              0FAD 3255      BRB 60$                    ; Release the channel.
              0FAF 3256      ENBINT                      ; Branch around.
              0FB2 3257 40$: BRW POWERFAIL              ; We had another POWERFAIL.
FF50 31      0FB5 3258 45$: SETIPL UCB$B_FIPL(R5)        ; Branch back to start again.
              0FB9 3259 50$: BRW TIMEOUT                ; We timed out on the SENSE.
FEE6 31      0FBC 3260 50$: SETIPL UCB$B_FIPL(R5)        ; Branch back to record TIMEOUT.
              0FBC 3261 60$: EXTZV #MF_NDTA_V_NDIC,-    ; We have the SENSE info.
              0FBC 3262 60$: #MF_NDTA_S_NDIC,-          ; Extract the interrupt code from
              0FBE 3263 60$: UCB$M_MF_NDTA_C(R5),R0     ; the status code returned as a result
50 0184 C5      0FBF 3264 60$: CMPB #HIC_DONE,R0        ; of the SENSE command.
50 01      0FC3 3265 60$: BEQL 65$                      ; See if SENSE was a success.
              0FC6 3266 60$: BRW 170$                   ; EQL => yes.
              0FC8 3267 60$: BRW 170$                   ; If not successful, branch to FATAL error.
              0FCB 3268 60$: MOVZWL UCB$M_MF_DS(R5),R2   ; R2 has SENSE info.
52 0158 C5      0FCD 3269 65$: BBS #MF_DS_V_REW,R2,75$  ; Branch if REWINDING.
21 52 0C      0FCE 3270 65$: BBS #MF_DS_V_ONL,R2,70$   ; Branch forward if offline.
04 52 0D      0FCE 3271 65$: BBS #MF_DS_V_BOT,R2,80$   ; Branch out of loop if at BOT.
1C 52 0A      0FCE 3272 70$: ACBB #15,#1,UCB$B_CEX(R5),20$ ; Loop 15 times between operator messages.
FF46 0093 C5      0FCE 3273 70$: MOVZBL #MSG$ DEVOFF[IN,R4 ; Get message number.
53 00000000'GF 0FCE 3274 70$: MOVAB G^SYS$GL_OPRMBX,R3  ; R3 => operator mailbox.
00000000'GF 16 0FCE 3275 70$: JSB G^EXE$SNDEVMSG      ; Send message to operator.
FF2E 31      0FCE 3276 75$: BRW 10$                    ; Go kill some more time.
              0FCE 3277 80$:
              0FCE 3278 80$:
              0FCE 3279 80$:
              0FCE 3280 80$:
              0FCE 3281 80$:
              0FCE 3282 80$:
              0FCE 3283 80$:
              0FCE 3284 80$:
              0FCE 3285 80$:
              0FCE 3286 80$:
              0FCE 3287 80$:
              0FCE 3288 80$:
              0FCE 3289 80$:
              0FCE 3290 80$:
              0FCE 3291 80$:
              0FCE 3292 80$:
              0FCE 3293 80$:
              0FCE 3294 80$:
              0FCE 3295 80$:
              0FCE 3296 80$:
              0FCE 3297 80$:
              0FCE 3298 80$:
              0FCE 3299 80$:
              0FCE 3300 80$:
              0FCE 3301 80$:
              0FCE 3302 80$:
              0FCE 3303 80$:
              0FCE 3304 80$:
              0FCE 3305 80$:
              0FCE 3306 80$:
              0FCE 3307 80$:
              0FCE 3308 80$:
              0FCE 3309 80$:
              0FCE 3310 80$:
              0FCE 3311 80$:
              0FCE 3312 80$:
              0FCE 3313 80$:
              0FCE 3314 80$:
              0FCE 3315 80$:
              0FCE 3316 80$:
              0FCE 3317 80$:
              0FCE 3318 80$:
              0FCE 3319 80$:
              0FCE 3320 80$:
              0FCE 3321 80$:
              0FCE 3322 80$:
              0FCE 3323 80$:
              0FCE 3324 80$:
              0FCE 3325 80$:
              0FCE 3326 80$:
              0FCE 3327 80$:
              0FCE 3328 80$:
              0FCE 3329 80$:
              0FCE 3330 80$:
              0FCE 3331 80$:
              0FCE 3332 80$:
              0FCE 3333 80$:
              0FCE 3334 80$:
              0FCE 3335 80$:
              0FCE 3336 80$:
              0FCE 3337 80$:
              0FCE 3338 80$:
              0FCE 3339 80$:
              0FCE 3340 80$:
              0FCE 3341 80$:
              0FCE 3342 80$:
              0FCE 3343 80$:
              0FCE 3344 80$:
              0FCE 3345 80$:
              0FCE 3346 80$:
              0FCE 3347 80$:
              0FCE 3348 80$:
              0FCE 3349 80$:
              0FCE 3350 80$:
              0FCE 3351 80$:
              0FCE 3352 80$:
              0FCE 3353 80$:
              0FCE 3354 80$:
              0FCE 3355 80$:
              0FCE 3356 80$:
              0FCE 3357 80$:
              0FCE 3358 80$:
              0FCE 3359 80$:
              0FCE 3360 80$:
              0FCE 3361 80$:
              0FCE 3362 80$:
              0FCE 3363 80$:
              0FCE 3364 80$:
              0FCE 3365 80$:
              0FCE 3366 80$:
              0FCE 3367 80$:
              0FCE 3368 80$:
              0FCE 3369 80$:
              0FCE 3370 80$:
              0FCE 3371 80$:
              0FCE 3372 80$:
              0FCE 3373 80$:
              0FCE 3374 80$:
              0FCE 3375 80$:
              0FCE 3376 80$:
              0FCE 3377 80$:
              0FCE 3378 80$:
              0FCE 3379 80$:
              0FCE 3380 80$:
              0FCE 3381 80$:
              0FCE 3382 80$:
              0FCE 3383 80$:
              0FCE 3384 80$:
              0FCE 3385 80$:
              0FCE 3386 80$:
              0FCE 3387 80$:
              0FCE 3388 80$:
              0FCE 3389 80$:
              0FCE 3390 80$:
              0FCE 3391 80$:
              0FCE 3392 80$:
              0FCE 3393 80$:
              0FCE 3394 80$:
              0FCE 3395 80$:
              0FCE 3396 80$:
              0FCE 3397 80$:
              0FCE 3398 80$:
              0FCE 3399 80$:
              0FCE 3400 80$:
              0FCE 3401 80$:
              0FCE 3402 80$:
              0FCE 3403 80$:
              0FCE 3404 80$:
              0FCE 3405 80$:
              0FCE 3406 80$:
              0FCE 3407 80$:
              0FCE 3408 80$:
              0FCE 3409 80$:
              0FCE 3410 80$:
              0FCE 3411 80$:
              0FCE 3412 80$:
              0FCE 3413 80$:
              0FCE 3414 80$:
              0FCE 3415 80$:
              0FCE 3416 80$:
              0FCE 3417 80$:
              0FCE 3418 80$:
              0FCE 3419 80$:
              0FCE 3420 80$:
              0FCE 3421 80$:
              0FCE 3422 80$:
              0FCE 3423 80$:
              0FCE 3424 80$:
              0FCE 3425 80$:
              0FCE 3426 80$:
              0FCE 3427 80$:
              0FCE 3428 80$:
              0FCE 3429 80$:
              0FCE 3430 80$:
              0FCE 3431 80$:
              0FCE 3432 80$:
              0FCE 3433 80$:
              0FCE 3434 80$:
              0FCE 3435 80$:
              0FCE 3436 80$:
              0FCE 3437 80$:
              0FCE 3438 80$:
              0FCE 3439 80$:
              0FCE 3440 80$:
              0FCE 3441 80$:
              0FCE 3442 80$:
              0FCE 3443 80$:
              0FCE 3444 80$:
              0FCE 3445 80$:
              0FCE 3446 80$:
              0FCE 3447 80$:
              0FCE 3448 80$:
              0FCE 3449 80$:
              0FCE 3450 80$:
              0FCE 3451 80$:
              0FCE 3452 80$:
              0FCE 3453 80$:
              0FCE 3454 80$:
              0FCE 3455 80$:
              0FCE 3456 80$:
              0FCE 3457 80$:
              0FCE 3458 80$:
              0FCE 3459 80$:
              0FCE 3460 80$:
              0FCE 3461 80$:
              0FCE 3462 80$:
              0FCE 3463 80$:
              0FCE 3464 80$:
              0FCE 3465 80$:
              0FCE 3466 80$:
              0FCE 3467 80$:
              0FCE 3468 80$:
              0FCE 3469 80$:
              0FCE 3470 80$:
              0FCE 3471 80$:
              0FCE 3472 80$:
              0FCE 3473 80$:
              0FCE 3474 80$:
              0FCE 3475 80$:
              0FCE 3476 80$:
              0FCE 3477 80$:
              0FCE 3478 80$:
              0FCE 3479 80$:
              0FCE 3480 80$:
              0FCE 3481 80$:
              0FCE 3482 80$:
              0FCE 3483 80$:
              0FCE 3484 80$:
              0FCE 3485 80$:
              0FCE 3486 80$:
              0FCE 3487 80$:
              0FCE 3488 80$:
              0FCE 3489 80$:
              0FCE 3490 80$:
              0FCE 3491 80$:
              0FCE 3492 80$:
              0FCE 3493 80$:
              0FCE 3494 80$:
              0FCE 3495 80$:
              0FCE 3496 80$:
              0FCE 3497 80$:
              0FCE 3498 80$:
              0FCE 3499 80$:
              0FCE 3500 80$:
              0FCE 3501 80$:
              0FCE 3502 80$:
              0FCE 3503 80$:
              0FCE 3504 80$:
              0FCE 3505 80$:
              0FCE 3506 80$:
              0FCE 3507 80$:
              0FCE 3508 80$:
              0FCE 3509 80$:
              0FCE 3510 80$:
              0FCE 3511 80$:
              0FCE 3512 80$:
              0FCE 3513 80$:
              0FCE 3514 80$:
              0FCE 3515 80$:
              0FCE 3516 80$:
              0FCE 3517 80$:
              0FCE 3518 80$:
              0FCE 3519 80$:
              0FCE 3520 80$:
              0FCE 3521 80$:
              0FCE 3522 80$:
              0FCE 3523 80$:
              0FCE 3524 80$:
              0FCE 3525 80$:
              0FCE 3526 80$:
              0FCE 3527 80$:
              0FCE 3528 80$:
              0FCE 3529 80$:
              0FCE 3530 80$:
              0FCE 3531 80$:
              0FCE 3532 80$:
              0FCE 3533 80$:
              0FCE 3534 80$:
              0FCE 3535 80$:
              0FCE 3536 80$:
              0FCE 3537 80$:
              0FCE 3538 80$:
              0FCE 3539 80$:
              0FCE 3540 80$:
              0FCE 3541 80$:
              0FCE 3542 80$:
              0FCE 3543 80$:
              0FCE 3544 80$:
              0FCE 3545 80$:
              0FCE 3546 80$:
              0FCE 3547 80$:
              0FCE 3548 80$:
              0FCE 3549 80$:
              0FCE 3550 80$:
              0FCE 3551 80$:
              0FCE 3552 80$:
              0FCE 3553 80$:
              0FCE 3554 80$:
              0FCE 3555 80$:
              0FCE 3556 80$:
              0FCE 3557 80$:
              0FCE 3558 80$:
              0FCE 3559 80$:
              0FCE 3560 80$:
              0FCE 3561 80$:
              0FCE 3562 80$:
              0FCE 3563 80$:
              0FCE 3564 80$:
              0FCE 3565 80$:
              0FCE 3566 80$:
              0FCE 3567 80$:
              0FCE 3568 80$:
              0FCE 3569 80$:
              0FCE 3570 80$:
              0FCE 3571 80$:
              0FCE 3572 80$:
              0FCE 3573 80$:
              0FCE 3574 80$:
              0FCE 3575 80$:
              0FCE 3576 80$:
              0FCE 3577 80$:
              0FCE 3578 80$:
              0FCE 3579 80$:
              0FCE 3580 80$:
              0FCE 3581 80$:
              0FCE 3582 80$:
              0FCE 3583 80$:
              0FCE 3584 80$:
              0FCE 3585 80$:
              0FCE 3586 80$:
              0FCE 3587 80$:
              0FCE 3588 80$:
              0FCE 3589 80$:
              0FCE 3590 80$:
              0FCE 3591 80$:
              0FCE 3592 80$:
              0FCE 3593 80$:
              0FCE 3594 80$:
              0FCE 3595 80$:
              0FCE 3596 80$:
              0FCE 3597 80$:
              0FCE 3598 80$:
              0FCE 3599 80$:
              0FCE 3600 80$:
              0FCE 3601 80$:
              0FCE 3602 80$:
              0FCE 3603 80$:
              0FCE 3604 80$:
              0FCE 3605 80$:
              0FCE 3606 80$:
              0FCE 3607 80$:
              0FCE 3608 80$:
              0FCE 3609 80$:
              0FCE 3610 80$:
              0FCE 3611 80$:
              0FCE 3612 80$:
              0FCE 3613 80$:
              0FCE 3614 80$:
              0FCE 3615 80$:
              0FCE 3616 80$:
              0FCE 3617 80$:
              0FCE 3618 80$:
              0FCE 3619 80$:
              0FCE 3620 80$:
              0FCE 3621 80$:
              0FCE 3622 80$:
              0FCE 3623 80$:
              0FCE 3624 80$:
              0FCE 3625 80$:
              0FCE 3626 80$:
              0FCE 3627 80$:
              0FCE 3628 80$:
              0FCE 3629 80$:
              0FCE 3630 80$:
              0FCE 3631 80$:
              0FCE 3632 80$:
              0FCE 3633 80$:
              0FCE 3634 80$:
              0FCE 3635 80$:
              0FCE 3636 80$:
              0FCE 3637 80$:
              0FCE 3638 80$:
              0FCE 3639 80$:
              0FCE 3640 80$:
              0FCE 3641 80$:
              0FCE 3642 80$:
              0FCE 3643 80$:
              0FCE 3644 80$:
              0FCE 3645 80$:
              0FCE 3646 80$:
              0FCE 3647 80$:
              0FCE 3648 80$:
              0FCE 3649 80$:
              0FCE 3650 80$:
              0FCE 3651 80$:
              0FCE 3652 80$:
              0FCE 3653 80$:
              0FCE 3654 80$:
              0FCE 3655 80$:
              0FCE 3656 80$:
              0FCE 3657 80$:
              0FCE 3658 80$:
              0FCE 3659 80$:
              0FCE 3660 80$:
              0FCE 3661 80$:
              0FCE 3662 80$:
              0FCE 3663 80$:
              0FCE 3664 80$:
              0FCE 3665 80$:
              0FCE 3666 80$:
              0FCE 3667 80$:
              0FCE 3668 80$:
              0FCE 3669 80$:
              0FCE 3670 80$:
              0FCE 3671 80$:
              0FCE 3672 80$:
              0FCE 3673 80$:
              0FCE 3674 80$:
              0FCE 3675 80$:
              0FCE 3676 80$:
              0FCE 3677 80$:
              0FCE 3678 80$:
              0FCE 3679 80$:
              0FCE 3680 80$:
              0FCE 3681 80$:
              0FCE 3682 80$:
              0FCE 3683 80$:
              0FCE 3684 80$:
              0FCE 3685 80$:
              0FCE 3686 80$:
              0FCE 3687 80$:
              0FCE 3688 80$:
              0FCE 3689 80$:
              0FCE 3690 80$:
              0FCE 3691 80$:
              0FCE 3692 80$:
              0FCE 3693 80$:
              0FCE 3694 80$:
              0FCE 3695 80$:
              0FCE 3696 80$:
              0FCE 3697 80$:
              0FCE 3698 80$:
              0FCE 3699 80$:
              0FCE 3700 80$:
              0FCE 3701 80$:
              0FCE 3702 80$:
              0FCE 3703 80$:
              0FCE 3704 80$:
              0FCE 3705 80$:
              0FCE 3706 80$:
              0FCE 3707 80$:
              0FCE 3708 80$:
              0FCE 3709 80$:
              0FCE 3710 80$:
              0FCE 3711 80$:
              0FCE 3712 80$:
              0FCE 3713 80$:
              0FCE 3714 80$:
              0FCE 3715 80$:
              0FCE 3716 80$:
              0FCE 3717 80$:
              0FCE 3718 80$:
              0FCE 3719 80$:
              0FCE 3720 80$:
              0FCE 3721 80$:
              0FCE 3722 80$:
              0FCE 3723 80$:
              0FCE 3724 80$:
              0FCE 3725 80$:
              0FCE 3726 80$:
              0FCE 3727 80$:
              0FCE 3728 80$:
              0FCE 3729 80$:
              0FCE 3730 80$:
              0FCE 3731 80$:
              0FCE 3732 80$:
              0FCE 3733 80$:
              0FCE 3734 80$:
              0FCE 3735 80$:
              0FCE 3736 80$:
              0FCE 3737 80$:
              0FCE 3738 80$:
              0FCE 3739 80$:
              0FCE 3740 80$:
              0FCE 3741 80$:
              0FCE 3742 80$:
              0FCE 3743 80$:
              0FCE 3744 80$:
              0FCE 3745 80$:
              0FCE 3746 80$:
              0FCE 3747 80$:
              0FCE 3748 80$:
              0FCE 3749 80$:
              0FCE 3750 80$:
              0FCE 3751 80$:
              0FCE 3752 80$:
              0FCE 3753 80$:
              0FCE 3754 80$:
              0FCE 3755 80$:
              0FCE 3756 80$:
              0FCE 3757 80$:
              0FCE 3758 80$:
              0FCE 3759 80$:
              0FCE 3760 80$:
              0FCE 3761 80$:
              0FCE 3762 80$:
              0FCE 3763 80$:
              0FCE 3764 80$:
              0FCE 3765 80$:
              0FCE 3766 80$:
              0FCE 3767 80$:
              0FCE 3768 80$:
              0FCE 3769 80$:
              0FCE 3770 80$:
              0FCE 3771 80$:
              0FCE 3772 80$:
              0FCE 3773 80$:
              0FCE 3774 80$:
              0FCE 3775 80$:
              0FCE 3776 80$:
              0FCE 3777 80$:
              0FCE 3778 80$:
              0FCE 3779 80$:
              0FCE 3780 80$:
              0FCE 3781 80$:
              0FCE 3782 80$:
              0FCE 3783 80$:
              0FCE 3784 80$:
              0FCE 3785 80$:
              0FCE 3786 80$:
              0FCE 3787 80$:
              0FCE 3788 80$:
              0FCE 3789 80$:
              0FCE 3790 80$:
              0FCE 3791 80$:
              0FCE 3792 80$:
              0FCE 3793 80$:
              0FCE 3794 80$:
              0FCE 3795 80$:
              0FCE 3796 80$:
              0FCE 3797 80$:
              0FCE 3798 80$:
              0FCE 3799 80$:
              0FCE 3800 80$:
              0FCE 3801 80$:
              0FCE 3802 80$:
              0FCE 3803 80$:
              0FCE 3804 80$:
              0FCE 3805 80$:
              0FCE 3806 80$:
              0FCE 3807 80$:
              0FCE 3808 80$:
              0FCE 3809 80$:
              0FCE 3810 80$:
              0FCE 3811 80$:
              0FCE 3812 80$:
              0FCE 3813 80$:
              0FCE 3814 80$:
              0FCE 3815 80$:
              0FCE 3816 80$:
              0FCE 3817 80$:
              0FCE 3818 80$:
              0FCE 3819 80$:
              0FCE 3820 80$:
              0FCE 3821 80$:
              0FCE 3822 80$:
              0FCE 3823 80$:
              0FCE 3824 80$:
              0FCE 3825 80$:
              0FCE 3826 80$:
              0FCE 3827 80$:
              0FCE 3828 80$:
              0FCE 3829 80$:
              0FCE 3830 80$:
              0FCE 3831 80$:
              0FCE 3832 80$:
              0FCE 3833 80$:
              0FCE 3834 80$:
              0FCE 3835 80$:
              0FCE 3836 80$:
              0FCE 3837 80$:
              0FCE 3838 80$:
              0FCE 3839 80$:
              0FCE 3840 80$:
              0FCE 3841 80$:
              0FCE 3842 80$:
              0FCE 3843 80$:
              0FCE 3844 80$:
              0FCE 3845 80$:
              0FCE 3846 80$:
              0FCE 3847 80$:
              0FCE 3848 80$:
              0FCE 3849 80$:
              0FCE 3850 80$:
              0FCE 3851 80$:
              0FCE 3852 80$:
              0FCE 3853 80$:
              0FCE 3854 80$:
              0FCE 3855 80$:
              0FCE 3856 80$:
              0FCE 3857 80$:
              0FCE 3858 80$:
              0FCE 3859 80$:
              0FCE 3860 80$:
              0FCE 3861 80$:
              0FCE 3862 80$:
              0FCE 3863 80$:
              0FCE 3864 80$:
              0FCE 3865 80$:
              0FCE 3866 80$:
              0FCE 3867 80$:
              0FCE 3868 80$:
              0FCE 3869 80$:
              0FCE 3870 80$:
              0FCE 3871 80$:
              0FCE 3872 80$:
              0FCE 3873 80$:
              0FCE 3874 80$:
              0FCE 3875 80$:
              0FCE 3876 80$:
              0FCE 3877 80$:
              0FCE 3878 80$:
              0FCE 3879 80$:
              0FCE 3880 80$:
              0FCE 3881 80$:
              0FCE 3882 80$:
              0FCE 3883 80$:
              0FCE 3884 80$:
              0FCE 3885 80$:
              0FCE 3886 80$:
              0FCE 3887 80$:
              0FCE 3888 80$:
              0FCE 3889 80$:
              0FCE 3890 80$:
              0FCE 3891 80$:
              0FCE 3892 80$:
              0FCE 3893 80$:
              0FCE 3894 80$:
              0FCE 3895 80$:
              0FCE 3896 80$:
              0FCE 3897 80$:
              0FCE 3898 80$:
              0FCE 3899 80$:
              0FCE 3900 80$:
              0FCE 3901 80$:
              0FCE 3902 80$:
              0FCE 3903 80$:
              0FCE 3904 80$:
              0FCE 3905 80$:
              0FCE 3906 80$:
              0FCE 3907 80$:
              0FCE 3908 80$:
              0FCE 3909 80$:
              0FCE 3910 80$:
              0FCE 3911 80$:
              0FCE 3912 80$:
              0FCE 3913 80$:
              0FCE 3914 80$:
              0FCE 3915 80$:
              0FCE 3916 80$:
              0FCE 3917 80$:
              0FCE 3918 80$:
              0FCE 3919 80$:
              0FCE 3920 80$:
              0FCE 3921 80$:
              0FCE 3922 80$:
              0FCE 3923 80$:
              0FCE 3924 80$:
              0FCE 3925 80$:
              0FCE 3926 80$:
              0FCE 3927 80$:
              0FCE 3928 80$:
              0FCE 3929 80$:
              0FCE 3930 80$:
              0FCE 3931 80$:
              0FCE 3932 80$:
              0FCE 3933 80$:
              0FCE 3934 80$:
              0FCE 3935 80$:
              0FCE 3936 80$:
              0FCE 3937 80$:
              0FCE 3938 80$:
              0FCE 3939 80$:
              0FCE 3940 80$:
              0FCE 3941 80$:
              0FCE 3942 80$:
              0FCE 3943 80$:
              0FCE 3944 80$:
              0FCE 3945 80$:
              0FCE 3946 80$:
              0FCE 3947 80$:
              0FCE 3948 80$:
              0FCE 3949 80$:
              0FCE 3950 80$:
              0FCE 3951 80$:
              0FCE 3952 80$:
              0FCE 3953 80$:
              0FCE 3954 80$:
              0FCE 3955 80$:
              0FCE 3956 80$:
              0FCE 3957 80$:
              0FCE 3958 80$:
              0FCE 3959 80$:
              0FCE 3960 80$:
              0FCE 3961 80$:
              0FCE 3962 80$:
              0FCE 3963 80$:
              0FCE 3964 80$:
              0FCE 3965 80$:
              0FCE 3966 80$:
              0FCE 3967 80$:
              0FCE 3968 80$:
              0FCE 3969 80$:
              0FCE 3970 80$:
              0FCE 3971 80$:
              0FCE 3972 80$:
              0FCE 3973 80$:
              0FCE 3974 80$:
              0FCE 3975 80$:
              0FCE 3976 80$:
              0FCE 3977 80$:
              0FCE 3978 80$:
              0FCE 3979 80$:
              0FCE 3980 80$:
              0FCE 3981 80$:
              0FCE 3982 80$:
              0FCE 3983 80$:
              0FCE 3984 80$:
              0FCE 3985 80$:
              0FCE 3986 80$:
              0FCE 3987 80$:
              0FCE 3988 80$:
              0FCE 3989 80$:
              0FCE 3990 80$:
              0FCE 3991 80$:
              0FCE 3992 80$:
              0FCE 3993 80$:
              0FCE 3994 80$:
              0FCE 3995 80$:
              0FCE 3996 80$:
              0FCE 3997 80$:
              0FCE 3998 80$:
              0FCE 3999 80$:
              0FCE 4000 80$:
              0FCE 4001 80$:
              0FCE 4002 80$:
              0FCE 4003 80$:
              0FCE 40
```

```
50 000000FF 8F D1 1016 3301 105$: CMPL #255,R0 ; See if greater than register limit.
      04 18 1016 3302 BGEQ 110$ ; If 255 > count, branch to use count.
      50 FF 8F 9A 101D 3303 MOVZBL #255,R0 ; If count > 255, use 255.
      1023 3304 110$:
      1023 3305 ASSUME MF_NDT0-S-CCNT EQ 8 ; Assume count field in register is
      1023 3306 ASSUME MF_NDT0-V-CCNT EQ 8 ; located in high byte.
      00BD C5 50 90 1023 3307 MOVBL R0,UCBSW MF_NDT0CR+1(R5) ; Complete command template with count.
      00BC C5 3C 1028 3308 MOVZWL UCBSW MF_NDT0CR(R5),- ; Record Hardware command about to
      0180 C5 102C 3309 UCBSW MF_CMD(R5) ; become the CURRENT command.
      102F 3310
      102F 3311 REQPCAN ; Sequentialize access to TM78.
      1035 3312 BWS #UCBSW MF_CNCLP - ; If CANCEL pending set while we were
      1035 3313 UCBSW_DEVSTS(R5),190$ ; resource waiting, branch around to end.
      103D 3314
      103D 3315
      103D 3316
      1043 3317 DSBINT ;
      104C 3318 PRIOR_TEST TMRDY ; Assure that TM78 ready.
      104E 3319 BBSC #UCBSW POWER, - ;
      1051 3320 UCBSW_STS(R5),120$ ; Forget it if another POWERFAIL.
      1055 3321 MOVZWL UCBSW_UNIT(R5),R0 ; Pickup unit number.
      1059 3322 MOVZWL UCBSW MF_NDT0CR(R5),- ; Move command template to
      105C 3323 MF_NDT0(R3)[R0] ; device register to initiate command.
      106A 3324 WFIKPC 130$,#TU78_MAX_SPACE ; Longest space should terminate in
      1073 3325 POST_TEST TMRDY ; Assure that TM78 ready.
      1073 3326 MOVL UCBSW MF_NDTA(R5), - ; less than 60 seconds.
      1077 3327 UCBSW MF_NDTA_C(R5) ; Copy NDT attention data for this
      107A 3328 BSBW SAVE_TM78_REGS ; interrupt to stable place.
      107D 3329 IOFORK ; Save hardware registers for ERROR LOG
      1083 3330 BBSC #UCBSW POWER, - ;
      1085 3331 UCBSW_STS(R5),125$ ; If another POWERFAIL, branch and clear.
      1088 3332 RELCHAN ; Else release the channel.
      108E 3333 BRB 140$ ; and branch around to continue.
      1090 3334 120$: ENBINT ; Enable after pre-POWERFAIL.
      1093 3335 125$: BRW 150$ ; Branch to restart entire rewind, etc.
      1096 3336
      1096 3337 130$: SETIPL UCBSW FIPL(R5) ; After TIMEOUT on space, lower IPL and
      109A 3338 BRW TIMEOUT ; Branch back to treat as TIMEOUT.
      109D 3339
      109D 3340 140$: EXTZV #MF_NDTA-V-NDIC,- ; Extract the interrupt code from
      109F 3341 #MF_NDTA-S-NDIC,- ; the status code returned as a result
      10A0 3342 UCBSW MF_NDTA_C(R5),R0 ; of the space command.
      10A4 3343 CMPB #HIC_DONE,R0 ; See if it is a success.
      10A7 3344 BEQL 160$ ; If EQL, then OK so branch.
      10A9 3345
      10A9 3346 CMPB #HIC_OFF_LINE,R0 ; See if drive has gone off-line.
      10AC 3347 BNEQ 170$ ; NEQ => some hardware error
      10AE 3348 150$: BRW POWERFAIL ; We had another POWERFAIL
      10AE 3349 ; so branch back to try again.
      10B1 3350
      10B1 3351
      10B1 3352 160$: ; Here we apparently successfully spaced
      10B1 3353 ; forward a group of records and TAPEMARKs.
      10B1 3354 ASSUME MF_NDT0-S-CCNT EQ 8 ;
      10B1 3355 ASSUME MF_NDT0-V-CCNT EQ 8 ; Assume command register format.
      10B1 3356 MOVZBL UCBSW MF_NDT0CR+1(R5),R0 ; Pickup number of places spaced over.
      10B6 3357 ADDL R0,UCBSW_RECORD(R5) ; Update position variable in UCB.
```

FF4B	31	10BB	3358	BRW	100\$	; Branch back to continue spacing.
		10BF	3359			
		10C0	3360		170\$:	; We had a fatal HARDWARE error.
02	AA	10C1	3361	BICW	#UCBSM MF POWER,-	
68 A5		10C0	3362		UCBSW_DEVSTS(R5)	; Clear bit to prevent RECURSIVE entry.
0800 8F	AA	10C2	3363	BICW	#UCBSM_VALID,-	
64 A5		10C6	3364		UCBSW_STS(R5)	; Indicate invalid volume.
		10C8	3365			
0102	30	10C8	3366	BSBW	DEVICE_ERROR	; Do EXTENDED SENSE and log device error
		10CB	3367			
50 008C 8F	3C	10CB	3368	MOVZWL	#SS\$_DRVERR,R0	; Indicate error status.
		10D0	3369	SUBRETURN		; Return to caller.
		10DA	3370			
		10DA	3371		180\$:	; Here we have successfully repositioned.
02	AA	10DA	3372	BICW	#UCBSM MF POWER,-	
68 A5		10DC	3373		UCBSW_DEVSTS(R5)	; Clear bit to prevent RECURSIVE entry.
10	AA	10DE	3374	BICW	#<MTSM_LOST@-16>,-	
46 A5		10E0	3375		UCBSL_DEVDEPEND+2(R5)	; Clear lost position bit.
0800 8F	A8	10E2	3376	BISW	#UCBSM_VALID,-	
64 A5		10E6	3377		UCBSW_STS(R5)	; Indicate VALID volume.
00CC C5	D0	10E8	3378	MOVL	UCBSL_MF_ORGPTM(R5),-	; Restore value of position of
00C8 C5		10EC	3379		UCBSL_MF_PREVTM(R5)	; previous TAPEMARK.
F040	31	10EF	3380	BRW	TF_RESTARTIO	; branch to restart I/O function.
		10F2	3381			
		10F2	3382		190\$:	; CANCEL pending flag noticed.
06	AA	10F2	3383	BICW	#UCBSM MF POWER!UCBSM_MF_CNCLP,-	
68 A5		10F4	3384		UCBSW_DEVSTS(R5)	; Clear flag signalling that we are
		10F6	3385			; repositioning AND that CANCEL was
		10F6	3386			; signalled.
0800 8F	AA	10F6	3387	BICW	#UCBSM_VALID,-	
64 A5		10FA	3388		UCBSW_STS(R5)	; Set volume software invalid.
50 2C	D0	10FC	3389	MOVL	#SS\$_ABORT,R0	; Set abort status.
0290	31	10FF	3390	BRW	FUNCTION_EXIT	; Branch to end function.

```
1102 3392      .SBTTL REWIND_ROUTINE
1102 3393
1102 3394      : REWIND_ROUTINE - routine to execute a hardware REWIND or UNLOAD command,
1102 3395      : wait for it to be done if bit IOSM_NOWAIT in R1 is clear, and
1102 3396      : return the hardware HIC code in R0. If we experience TIMEOUT or
1102 3397      : POWERFAIL and the TIMEOUT_POWERFAIL returns to us we take the
1102 3398      : error exit. Else we return to 2 beyond the calling address.
1102 3399
1102 3400      : INPUTS:
1102 3401      : R0 = Hardware command and GO_BIT (REWIND or UNLOAD)
1102 3402      : R1 = Bitmask whose only bit of interest is IOSM_NOWAIT
1102 3403      : R3 => TM78_CSR
1102 3404      : R5 => UCB
1102 3405
1102 3406      : OUTPUTS:
1102 3407      : If no error exit.
1102 3408      : R0 = HIC code returned by hardware.
1102 3409
1102 3410
1102 3411 REWIND_ROUTINE:
1102 3412
1102 3413      SUBSAVE
1102 3414      SUBPUSH R1                      ; Save bitmask.
1102 3415
1102 3416      EX_NDT_CMD      CMD=R0,-          ; Command is in R0.
1102 3417      TIMEOUT=#TU78_MAX_REWIND,-; It may take a while.
1102 3418      ERROR_LABEL=30$      ; Where to go if NO GOOD.
1102 3419
1102 3420      :
1102 3421      : EX_NDT_CMD returns HIC in R0.
1102 3422      :
1102 3423
1102 3424      SUBPOP R1                      ; Restore bitmask.
1102 3425      CMPB #HIC_REWINDING,R0        ; Did we get REWINDING interrupt?
1102 3426      B_WNEQ 50$                   ; If NOT, branch around.
1102 3427      MOVZBL #HIC_DONE,R0          ; If YES, change it to DONE for now.
1102 3428
1102 3429      DSBINT                      ; If REWINDING, then block interrupts
1102 3430      PRIOR_TEST TMRDY            ; Assure that TM78 ready.
1102 3431      BBS #UCBSV_POWER,-          ; And then if IOSV_NOWAIT not specified
1102 3432      UCBSW_STS(R5),10$           ; then wait for DONE interrupt unless
1102 3433      BBS #IOSV_NOWAIT,R1,40$      ; it has already occurred. Here we test
1102 3434      ; IOSV_NOWAIT.
1102 3435      BBC #UCBSV_MF_REWIND,-       ; And here we test if DONE has already
1102 3436      UCBSW_DEVSTS(R5),40$        ; occurred.
1102 3437
1102 3438      WFIKPBH 20$,#TU78_MAX_REWIND ; Here we wait for the DONE interrupt.
1102 3439      POST_TEST TMRDY            ; Assure that TM78 ready.
1102 3440      BSBW TF_UNSOINT            ; Record data after interrupt after
1102 3441      ; REWIND supposedly done.
1102 3442      MOVL UCBSL_MF_NDTA(R5),-     ; Copy NDT attention data for this
1102 3443      UCBSL_MF_NDTA(C(R5))        ; interrupt to stable place.
1102 3444      BSBW SAVE_TM78_REGS         ; Save Hardware registers.
1102 3445
1102 3446      IOFORK
1102 3447      BBS #UCBSV_POWER,-          ; Test for the last time for
1102 3448      UCBSW_STS(R5),20$          ; POWERFAIL.
```

50	07	91	1133	3425	CMPB #HIC_REWINDING,R0	
			1136	3426	B_WNEQ 50\$	
50	01	9A	113B	3427	MOVZBL #HIC_DONE,R0	
			113E	3428		
			113E	3429	DSBINT	
			1144	3430	PRIOR_TEST TMRDY	
	05	E0	114D	3431	BBS #UCBSV_POWER,-	
41 64	A5		114F	3432	UCBSW_STS(R5),10\$	
65 51	07	E0	1152	3433	BBS #IOSV_NOWAIT,R1,40\$	
			1156	3434		
	00	E1	1156	3435	BBC #UCBSV_MF_REWIND,-	
60 68	A5		1158	3436	UCBSW_DEVSTS(R5),40\$	
			115B	3437		
			115B	3438	WFIKPBH 20\$,#TU78_MAX_REWIND	
			1169	3439	POST_TEST TMRDY	
04C0		30	1172	3440	BSBW TF_UNSOINT	
			1175	3441		
0168 C5		D0	1175	3442	MOVL UCBSL_MF_NDTA(R5),-	
0184 C5			1179	3443	UCBSL_MF_NDTA(C(R5))	
05CF		30	117C	3444	BSBW SAVE_TM78_REGS	
			117F	3445		
			117F	3446	IOFORK	
	05	E0	1185	3447	BBS #UCBSV_POWER,-	
0C 64	A5		1187	3448	UCBSW_STS(R5),20\$	

```
00 EF 118A 3449 EXTZV #MF_NDTA_V_NDIC,- ; Here we extract the interrupt code
06 118C 3450 ; from the register copy deposited
50 0184 C5 118D 3451 UCB$M_MF_NDTA_C(R5),R0 ; in the UCB above.
      1191 3452
      2B 11 1191 3453 BRB 50$ ; Branch around.
      1193 3454
      1193 3455 10$: ENBINT ; Enable interrupts after POWER bit set.
      1196 3456 20$: SETIPL UCB$B_FIPL(R5) ; Lower IPL if TIMEOUT.
FCD9 30 119A 3457 BSBW TIMEOUT_POWERFAIL ; We had a TIMEOUT or a POWERFAIL, call
      119D 3458 ; to handle it.
      OA 11 119D 3459 BRB 35$ ; Branch around the label where the
      119F 3460 ; error path from above EX_NDT_CMD
      119F 3461 ; rejoins the normal flow.
      119F 3462
      119F 3463 30$: ; We are here after experiencing
      119F 3464 ; an error return from EX_NDT_CMD.
      119F 3465 SUBPOP R1 ; Cleanup bitmask left on stack.
      11A9 3466 35$: ; Here we had an error in EX_NDT_CMD
      11A9 3467 ; or we experienced return from
      11A9 3468 ; TIMEOUT_POWERFAIL routine. Either
      11A9 3469 ; way we want to effect an error
      11A9 3470 ; return to our caller.
      11A9 3471
      7E 61 32 11A9 3472 SUBPOP R1 ; R1 => our return point.
      11B3 3473 CVTWL (R1),-(SP) ; Put relative addr of error return onto
      11B6 3474 ; the top of the stack.
      51 8E C0 11B6 3475 ADDL (SP)+,R1 ; Add in absolute offset. R1 => error return
      OD 11 11B9 3476 BRB 60$ ; Branch around
      11BB 3477 40$:
      11BB 3478 ENBINT
      11BE 3479 50$:
      11BE 3480 SUBPOP R1 ; R1 => return point.
      11C8 3481 60$:
      51 02 C0 11C8 3482 ADDL #2,R1 ; Adjust to correct return or error address
      61 17 11CB 3483 JMP (R1) ; Jump to return or ERROR_LABEL.
```

```
11CD 3485 .SBTTL Device Error Routine.
11CD 3486
11CD 3487 ; DEVICE_ERROR - routine to interface to INVOKE_EXTENDED_SENSE and the
11CD 3488 ; ERROR_LOGGER.
11CD 3489 ;
11CD 3490
11CD 3491
11CD 3492 DEVICE_ERROR:
11CD 3493
11CD 3494 SUBSAVE
11D7 3495 SUBPUSH R0,R1,R2 ; Save volatile registers.
07 68 08 E5 11F5 3496 BBCC #UCB$V_MF_SOFT_ERR,- ; If we NOT only a soft error, branch
07 68 A5 11F7 3497 UCB$W_DEVSTS(R5),10$ ; around. Else clear bit.
0F 44 0E E1 11FA 3498 BBC #MT$V_LOGSOFT,- ; If only soft error, branch around if
0F 44 A5 11FC 3499 UCB$L_DEVDEPEND(R5),30$ ; soft logging not enabled.
04 11 11FF 3500 BRB 20$ ; Soft logging enabled, branch around
1201 3501 ; decrementing of retry count.
0080 C5 97 1201 3502 10$: DECB UCB$B_ERTCNT(R5) ; Indicate another error has occurred
1205 3504 ; on this I/O operation.
1205 3505 20$:
002E 30 1205 3506 BSBW INVOKE_EXTENDED_SENSE
00000000'GF 16 1208 3507 JSB G^ERL$DEVICERR ; Invoke ERROR_LOGGER.
120E 3508 30$:
120E 3509 SUBPOP R2,R1,R0 ; Restore registers.
122C 3510 SUBRETURN
```



```
1236 3512 .SBTTL Invoke Extended Sense routine
1236 3513
1236 3514 ; INVOKE_EXTENDED_SENSE - subroutine used to issue an EXTENDED SENSE command.
1236 3515 ;
1236 3516 : INPUTS:
1236 3517 : R5 => UCB
1236 3518 :
1236 3519 : If the channels are owned upon entry then
1236 3520 : R3 => TM78 CSR
1236 3521 : R4 => MBA CSR
1236 3522 : else R3 and R4 have random junk that must be preserved.
1236 3523 :
1236 3524 : OUTPUTS:
1236 3525 : No specific outputs.
1236 3526 :
1236 3527 : SIDE EFFECTS:
1236 3528 : The results of the EXTENDED SENSE command are left in the UCB.
1236 3529 :
1236 3530 : Registers R0 - R2 are destroyed.
1236 3531 :
1236 3532 :
1236 3533 INVOKE_EXTENDED_SENSE:
1236 3534
1236 3535 BBC #UCBSV MF EXSNS DONE,- ; Never do more than one EXTENDED
1238 3536 UCB$W_DEVSTS(R5),10$ ; SENSE per I/O operation.
1238 3537 RSB ; So RETURN if already done.
123C 3538 10$:
123C 3539 BISW #UCBSM MF EXSNS DONE,- ; Set bit to indicate that we
1240 3540 UCB$W_DEVSTS(R5) ; are doing an EXTENDED SENSE.
1242 3541
124C 3542 SUBSAVE
1256 3543 SUBPUSH R3
1260 3544 SUBPUSH R4
1260 3545
1260 3545 MOVL UCB$L_CRB(R5),R0 ; R0 => TM78 CRB.
1264 3546 MOVL CRB$L_INTD+VEC$L_IDB(R0),R0 ; R0 => TM78 IDB.
1268 3547 CMPL R5,IDB$L_OWNER(R0) ; Are we channel owner?
126C 3548 BEQL 20$ ; EQL implies yes.
126E 3549
126E 3550 REQPCN ; Returns R4 => TM78 CSR.
1274 3551 MOVL R4,R3 ; Copy so R3 => TM78 CSR.
1277 3552 BRB 30$ ; Branch around.
1279 3553 20$:
1279 3554 BISW #UCBSM MF OWNPCN,- ; Set bit to indicate ownership.
127D 3555 UCB$W_DEVSTS(R5)
127F 3556 30$:
127F 3557 MOVL UCB$L_CRB(R5),R0 ; R0 => TM78 CRB.
1283 3558 MOVL CRB$L_LINK(R0),R0 ; R0 => MBA CRB.
1287 3559 MOVL CRB$L_INTD+VEC$L_IDB(R0),R0 ; R0 => MBA IDB.
128B 3560 CMPL R5,IDB$L_OWNER(R0) ; Are we channel owner?
128F 3561 BEQL 40$ ; EQL implies yes.
1291 3562
1291 3563 REQSCN ; R4 => MBA CSR.
1297 3564 BRB 50$ ; Branch around.
1299 3565 40$:
1299 3566 BISW #UCBSM MF OWNSCN,- ; Set bit to indicate ownership.
129D 3567 UCB$W_DEVSTS(R5)
129F 3568 50$:
```

```

      129F 3569      ASSUME UCB$S_SVAPTE+4 EQ UCB$W_BOFF
      129F 3570      ASSUME UCB$W_BOFF+2 EQ UCB$W_BCNT
      78 A5 7D 129F 3571      MOVQ UCB$S_SVAPTE(R5),-
00D8 C5      12A2 3572      UCB$Q_MF_TEMP(R5) ; Save data in temporary.
      12A5 3573      SUBPUSH UCB$W_MF_CS1(R5) ; Ditto.
      12B1 3574
      12B1 3575
      12B1 3576      ASSUME UCB$S_MF_SVAPTE+4 EQ UCB$W_MF_BOFF
      12B1 3577      ASSUME UCB$W_MF_BOFF+2 EQ UCB$W_MF_BCNT
      12B1 3578
00D0 C5 7D 12B1 3579      MOVQ UCB$S_MF_SVAPTE(R5),- ; Copy parameters needed to load MBA
      78 A5 12B5 3580      UCB$S_SVAPTE(R5) ; registers to effect EXTENDED SENSE.
      3B 9B 12B7 3581      MOVZBW #F_EXSNS!GO_BIT,-
00B4 C5 12B9 3582      UCB$W_MF_CST(R5) ; Indicate command about to invoke.
      FAE9 30 12BC 3583      BSBW LOAD_MBA_REGISTERS
00B8 C5 3C 12BF 3584      MOVZWL UCB$W_MF_TMPLTC(R5),- ; Copy TEMPLATE TC to get CMD ADR field
      08 A3 12C3 3585      MF_TC(R3) ; into hardware register. Other fields
      12C5 3586 ; are ignored anyway.
      12C5 3587
      12C5 3588
      12CB 3589      DSBINT
1C 68 A5 E0 12CD 3590      BBS #UCB$V_MF_ATTN,- ; If we got another 'B' type fault,
      05 E0 12D0 3591      UCB$W_DEVSTS(R5),60$ ; then branch around the EXTENDED SENSE
      17 64 A5 E0 12D2 3592      BBS #UCB$V_POWER,-
00B4 C5 3C 12D5 3593      UCB$W_STS(R5),60$ ; Branch around if we had POWERFAIL.
      63 12D9 3594      MOVZWL UCB$W_MF_CS1(R5),-
      12DA 3595      MF_CST(R3) ; Move command to hardware register.
      12E4 3596      WFIKPC 70$,#MINIMUM_TIMEOUT ; Wait.
      07 11 12EA 3597      IOFORK
      12EC 3598 60$:      BRB 80$
      12EF 3599 70$:      ENBINT
      12F3 3600 80$:      SETIPL UCB$B_FIPL(R5) ; Lower IPL after timeout.
      12F3 3601
      12F3 3602
      12F3 3603 ; Note - We come here under all circumstances. Either the command worked,
      12F3 3604 ; it didn't work, we had a POWERFAIL or we had a timeout. In the
      12F3 3605 ; first case (i.e. the command worked) we have the EXTENDED SENSE
      12F3 3606 ; data in the UCB. In the other cases we do not have it and we lose
      12F3 3607 ; all information as to why the command failed.
      12F3 3608
      12F3 3609
      00D8 C5 7D 12FF 3610      SUBPOP UCB$W_MF_CS1(R5) ; Restore
      78 A5 1303 3611      MOVQ UCB$Q_MF_TEMP(R5),-
      0B E4 1305 3612      UCB$S_SVAPTE(R5) ; Restore.
06 68 A5 1307 3613      BBSC #UCB$V_MF_OWNCHN,-
      130A 3614      UCB$W_DEVSTS(R5),90$ ; Branch around if we owned the channel.
      1310 3615      RELSCHN ; Release channel obtained only for
      1310 3616 ; EXTENDED SENSE.
      1310 3617 90$:
06 68 A5 E4 1310 3618      BBSC #UCB$V_MF_OWNPCHN,-
      1312 3619      UCB$W_DEVSTS(R5),100$ ; Branch around if we owned the channel.
      1315 3620      RELCHAN ; Release channel obtained only for
      1318 3621 ; EXTENDED SENSE.
      1318 3622 100$:
      1318 3623      SUBPOP R4
      1325 3624      SUBPOP R3
      132F 3625
```

TFDRIVER
V04-000

- TM78/TU78 MAGTAPE DRIVER
Invoke Extended Sense routine

J 10

16-SEP-1984 00:08:15 VAX/VMS Macro V04-00
5-SEP-1984 00:17:37 [DRIVER.SRC]TFDRIVER.MAR;1

Page 80
(1)

TF
VC

132F 3626 SUBRETURN

```
1339 3628 .SBTTL AWAIT_MANUAL_REWIND
1339 3629
1339 3630 : AWAIT_MANUAL_REWIND
1339 3631 :
1339 3632 : INPUTS:
1339 3633 :
1339 3634 : We are at device IPL following a WFIKPCH.
1339 3635 : We have just received an HIC NOT_RDY interrupt code in response
1339 3636 : to a hardware command implying that the drive is undergoing a
1339 3637 : manual REWIND or LOAD operation. What we do here is simply wait
1339 3638 : for this MANUAL operation to finish before trying to start the
1339 3639 : I/O operation that was requested.
1339 3640 :
1339 3641 : We may currently own the primary channel and the secondary channel.
1339 3642 :
1339 3643 : R3 => TM78 CSR
1339 3644 : R5 => UCB
1339 3645 :
1339 3646 :
1339 3647 .ENABLE LSB
1339 3648 AWAIT_MANUAL_REWIND:
1339 3649
1339 3650 IOFORK ; Lower IPL.
1339 3651 RELCHAN ; Release all channels.
1345 3652 DSBINT ; Block interrupts
1348 3653 PRIOR_TEST_TMRDY ; Assure that TM78 ready.
1354 3654
27 64 05 E0 1354 3655 BBS #UCB$V POWER,- ; Branch if there has been a POWERFAIL
1356 3656 UCB$W STS(R5),10$
2E 68 00 E1 1359 3657 BBC #UCB$V MF_REWIND,- ; Branch around WAIT if REWIND done.
1358 3658 UCB$W DEVSTS(R5),30$ ; Wait for REWIND, LOAD or DSE to finish.
135E 3659 WFIKPCH 20$,#TU78_MAX_REWIND ; Label of return point for WFIKPCH.
136C 3660 WAITING_FOR_MANUAL: ; Assure that TM78 ready.
136C 3661 POST_TEST_TMRDY ; Update UCB after REWIND or LOAD.
02BD 30 1375 3662 BSBW TF_UNRESOLNT
OF 11 137E 3664 BRB 40$ ; Branch around.
1380 3665
1380 3666 10$: ENBINT
1383 3667 20$: SETIPL UCB$B FIPL(R5) ; Set to fork IPL.
FAEC 30 1387 3668 BSBW TIMEOUT POWERFAIL ; Call to indicate problem.
06 11 138A 3669 BRB FUNCTION_EXIT ; If we experience return, goto EXIT.
138C 3670 30$: ENBINT ; Enable interrupts if REWIND done.
138C 3671 40$:
138F 3672
138F 3673
EDA0 31 138F 3674 BRW TF_RESTARTIO ; Goto restart I/O operation.
1392 3675 .DISABLE LSB
```

```
1392 3677 .SBTTL Function Completion Common Exit
1392 3678
1392 3679 : FUNCTION_EXIT - We branch here to terminate processing of a request.
1392 3680 :
1392 3681 : INPUTS:
1392 3682 :
1392 3683 : R0 = Final I/O completion status in the low word, # bytes transfered
1392 3684 : or records/files skipped in the high word.
1392 3685 :
1392 3686 : R5 => UCB
1392 3687 :
1392 3688 FUNCTION_EXIT:
1392 3689
1392 3690 PUSHL R0 ; Save final status and count.
1394 3691 JSB G^IOCS$DIAGBUFILL ; Fill diagnostic buffer if present.
139A 3692 BLBS (SP),20$ ; LBS implies successful completion.
139D 3693 MOVL UCBSL_IRP(R5),R4 ; R4 => current I/O packet.
13A1 3694 BBC #IRPSV_VIRTUAL,- ;
13A3 3695 IRPSW_STS(R4),20$ ; If CLEAR, NOT virtual function.
13A6 3696 MOVL IRPSL_WIND(R4),R4 ; R4 => Window Block.
13AA 3697 CLRW WCB$W_NMAP(R4) ; Clear number of mapping pointers.
13AD 3698 MOVL UCBSL_VCB(R5),R4 ; R4 => VCB listhead.
13B1 3699 MOVAB UCBSL_IOQFL(R5),R2 ; R2 => I/O Queue.
13B5 3700 MOVL R2,R3 ; Initialize R3 => previous entry.
13B8 3701 10$: MOVL (R3),R3 ; R3 => next entry.
13BB 3702 CMPL R3,R2 ; End of list?
13BE 3703 BEQL 20$ ; EQL implies yes, end of list.
13C0 3704 BBC #IRPSV_VIRTUAL,- ;
13C2 3705 IRPSW_STS(R3),10$ ; Clear implies NOT virtual function.
13C5 3706 MOVL 4(R3),R3 ; Retrieve address of previous entry.
13C9 3707 REMQUE @R3,R1 ; Remove entry from Driver queue.
13CD 3708 INSQUE (R1),@4(R4) ; Insert entry in blocked I/O list.
13D1 3709 BRB 10$ ; Loop back.
13D3 3710
13D3 3711 20$: POPL R0 ; Retrieve final I/O status.
13D6 3712 STSXIT:
13D6 3713 DSBINT ; No interrupts while we decide whether
13DC 3714 BBC #UCBSV_MF_ATTN,- ; to clear UCBFREE.
13DE 3715 UCBSW_DEVSTS(R5),30$ ; If no 'B' type faults pending, branch
13E1 3716 ENBINT ; around.
13E4 3717 MOVAB UCBSL_MF_SUBSTACK(R5),- ; Re-enable interrupts.
13E8 3718 UCBSL_MF_SUBSP(R5) ; Initialize SUBSTACK pointer in case we
13EB 3719 SUBPUSH R0 ; go here via CANCEL_IO.
13F5 3720 BSBW ISSUE_TMCLR ; Save final I/O status.
13F8 3721 SUBPOP R0 ; Go try to reset TM78.
1402 3722 DSBINT ; Restore final I/O status.
1408 3723 30$: ; Disable to match with coming ENBINT.
1408 3724 BISW #UCBSM_MF_UCBFREE,- ; UCB fork block now free for asynchronous
140C 3725 UCBSW_DEVSTS(R5) ; use.
140E 3726 ENBINT ; Re-enable interrupts.
1411 3727 MOVL UCBSL_DEVDEPEND(R5),R1 ; Set magtape status and characteristics.
1415 3728 REQCOM ; Complete I/O request.
1415 3729
```

```
141B 3731 .SBTTL TM78/TU78 REGISTER DUMP ROUTINE
141B 3732 :
141B 3733 : TF_REGDUMP - TM78/TU78 REGISTER DUMP ROUTINE
141B 3734 :
141B 3735 : THIS ROUTINE IS CALLED TO SAVE THE CONTROLLER AND DRIVE REGISTERS IN A
141B 3736 : SPECIFIED BUFFER. IT IS CALLED FROM THE DEVICE ERROR LOGGING ROUTINE AND
141B 3737 : FROM THE DIAGNOSTIC BUFFER FILL ROUTINE.
141B 3738 :
141B 3739 : INPUTS:
141B 3740 :
141B 3741 : R0 = ADDRESS OF REGISTER SAVE BUFFER.
141B 3742 : R4 = ADDRESS OF ADAPTER CONFIGURATION REGISTER.
141B 3743 : R5 = DEVICE UNIT UCB ADDRESS.
141B 3744 :
141B 3745 : OUTPUTS:
141B 3746 :
141B 3747 : THE CONTROLLER AND DRIVE REGISTERS ARE SAVED IN THE SPECIFIED BUFFER.
141B 3748 :
141B 3749 :
141B 3750 TF_REGDUMP: ; TM78/TU78 REGISTER DUMP ROUTINE
141B 3751 PUSHF ; Save registers destroyed by MOVC3.
141B 3752 MOVL #UCB_REGDUMP_LEN/4,(R0)+ ; Copy number of longwords.
1420 3753 MOVC3 #UCB_REGDUMP_LEN,- ; Copy previously saved device and
1424 3754 UCB$[MF MBA CSR(R5),(R0) ; MBA register values and other data.
1428 3755 POPR #^M<R0,R1,R2,R3,R4,R5> ; Restore registers destroyed by MOVC3.
142A 3756 RSB ;
```

60 80 00A4 0120 3F 29 8F C5 3F BA 05 BB D0 2E


```

46 19 1498 3815 BLSS 30$ ; do not allow configuration of device
      149A 3816
40 A4 000027FE 8F D0 149A 3817 MOVL #^023776, MF_ID(R4) ; Check the fifth location
      50 44 A4 D0 14A2 3818 MOVL MF_ID(R4), R0 ; Get the revision number
      02 50 91 14A6 3819 CMPB R0, #REV_LOC_4 ; If # is less then expected then
      35 19 14A9 3820 BLSS 30$ ; do not allow configuration of device
      14AB 3821
40 A4 00002FFE 8F D0 14AB 3822 MOVL #^027776, MF_ID(R4) ; Check the sixth location
      50 44 A4 D0 14B3 3823 MOVL MF_ID(R4), R0 ; Get the revision number
      03 50 91 14B7 3824 CMPB R0, #REV_LOC_5 ; If # is less then expected then
      24 19 14BA 3825 BLSS 30$ ; do not allow configuration of device
      14BC 3826
40 A4 000037FE 8F D0 14BC 3827 MOVL #^033776, MF_ID(R4) ; Check the seventh location
      50 44 A4 D0 14C4 3828 MOVL MF_ID(R4), R0 ; Get the revision number
      08 50 91 14C8 3829 CMPB R0, #REV_LOC_6 ; If # is less then expected then
      13 19 14CB 3830 BLSS 30$ ; do not allow configuration of device
      14CD 3831
40 A4 00003FFE 8F D0 14CD 3832 MOVL #^037776, MF_ID(R4) ; Check the eighth location
      50 44 A4 D0 14D5 3833 MOVL MF_ID(R4), R0 ; Get the revision number
      03 50 91 14D9 3834 CMPB R0, #REV_LOC_7 ; If # is less then expected then
      02 19 14DC 3835 BLSS 30$ ; do not allow configuration of device
      21 11 14DE 3836 BRB 40$
      14E0 3837
      38 BB 14E0 3838 30$: PUSHB #^M<R3,R4,R5> ; Save registers
      55 04 A6 D0 14E2 3839 MOVL DDB$L_UCB(R6), R5 ; Get first UCB address
54 00000000 8F D0 14E6 3840 MOVL #MSG$-TM78MVER, R4 ; Set message number in R4
53 00000000 GF 9E 14ED 3841 MOVAB G^SYS$GL_OPRMBX, R3 ; R3 => operators mailbox
      00000000 GF 16 14F4 3842 JSB G^EXE$SNDEVMSG ; Call to send oper message
      38 BA 14FA 3843 POPR #^M<R3,R4,R5> ; Restore registers
      03 11 14FC 3844 BRB 40$ ; For FT1 allow device to conf
      002D 31 14FE 3845 BRW 70$ ; Do not allow device to configure
      1501 3846
44 A4 00000100 8F CA 1501 3847 40$: BICL #MF_ID_M_HOLD, MF_ID(R4) ; Clear hold bit
44 A4 00004000 8F C8 1509 3848 BICL #MF_ID_M_TMCLR, MF_ID(R4) ; Start controller clear.
      50 FA 8F 9A 1511 3849 MOVZBL #250, R0 ; Count to register.
      FD 50 F5 1515 3850 50$: SOBGTR R0, 50$ ; Waste some time as TM78
      1518 3851 ; settles down after clear.
      1518 3852
50 001E8480 8F D0 1518 3853 MOVL #2000000, R0 ; Count to register again.
      151F 3854 60$:
      44 A4 DD 151F 3855 PUSHB MF_ID(R4) ; Push TM78 register to stack.
8E 00008000 8F D3 1522 3856 BITL #MF_ID_M_TMRDY, (SP)+ ; See if TM78 has come READY.
      03 12 1529 3857 BNEQ 70$ ; NEQ implies READY, so branch.
      F1 50 F5 152B 3858 SOBGTR R0, 60$ ; Loop waiting for device READY
      152E 3859 70$:
      50 8ED0 152E 3860 POPL R0 ; Restore register.
      05 1531 3861 RSB ; Return to caller.
```



```
1532 3863 .SBTTL TM78-TU78 TAPE DRIVE INITIALIZATION
1532 3864 :
1532 3865 : TM78_TXXX_INIT - TM78-TU78 TAPE DRIVE INITIALIZATION
1532 3866 :
1532 3867 : THIS ROUTINE IS CALLED AT SYSTEM INITIALIZATION AND AT POWER RECOVERY TO SET
1532 3868 : DRIVE PARAMETERS.
1532 3869 :
1532 3870 : INPUTS:
1532 3871 :
1532 3872 : R3 = ADDRESS OF TM78 DRIVE CONTROL REGISTER.
1532 3873 : R4 = ADDRESS OF MBA CONFIGURATION STATUS REGISTER.
1532 3874 : R5 = DEVICE UNIT UCB ADDRESS.
1532 3875 :
1532 3876 : OUTPUTS:
1532 3877 :
1532 3878 : UNIT PARAMETERS ARE ESTABLISHED.
1532 3879 :
1532 3880 :
1532 3881 TM78_TXXX_INIT: ;TU78 TAPE DRIVE INITIALIZATION
1532 3882 SOBL3 R4,R3,R2 ;CALCULATE OFFSET TO DRIVE CONTROL REGISTER
1536 3883 MOVAB -MBA$$_ERB(R2),R2 ;SUBTRACT OUT EXTERNAL REGISTER BASE
1538 3884 DIVW3 #127,R2,UCB$$_SLAVE(R5) ;SET ADAPTER DRIVE NUMBER
1543 3885 MULB3 #127/4,UCB$$_SLAVE(R5),UCB$$_SLAVE+1(R5) ;SET DRIVE OFFSET CONSTANT
1548 3886
1548 3887 PUSHL MBA$$_SR(R4) ;READ MBA STATUS REGISTER
154E 3888
154E 3889 BwBS #UCB$$_POWER,- ; Branch around if here at POWERFAIL.
154E 3890 UCB$$_STS(R5),50$
1556 3891 :
1556 3892 : If the following bit is not set then the TM78 ucode is not at the latest rev.
1556 3893 : The integrity of the drive can not be guaranteed if the device is not up
1556 3894 : to rev, hence do not configure the device until it is been brought up
1556 3895 : to rev. This will be true in a release after after the ucode is available
1556 3896 : however until that time we will the device to be configured into the system.
1556 3897 : We will also send a message to the operators console informing the user
1556 3898 : of the problem.
1556 3899 :
1556 3900 PUSHL MF_ID(R3) ; Get the indirect register
1559 3901 BITL #MF_ID_M_HOLD,(SP)+ ; If NEQ then bit is set do not
1560 3902 BwNEQ 50$ ; configure units (ucode not up to rev)
1565 3903 BISW #UCB$$_MF_UCBFREE,- ; Initialize bit that indicates the UCB
1569 3904 UCB$$_DEVSTS(R5) ; fork block is free for asynchronous
1568 3905 ; use.
1568 3906
1568 3907 MOVZBL UCB$$_SLAVE(R5),R1 ; R1 = MBA port number for TM78.
1570 3908 ASHL R1,#1,R1 ; R1 = mask corresponding to attention bit.
1574 3909 MOVL R1,MF_AB(R3) ; Clear it just in case set.
1578 3910
1578 3911 MOVZBL #250,R2 ; Set a counter with small number
157C 3912 SOBGR R2,5$ ; Waste time while ATTN bit clears.
157F 3913
157F 3914 MOVZWL UCB$$_UNIT(R5),R2 ; R2 = unit number of this drive.
1583 3915 MOVZBL #F_SENSE!GO_BIT,- ; Do a SENSE command to see what kind
1585 3916 MF_NDT0(R3)[R2] ; drive we have.
1588 3917 MOVL #2000000,R2 ; Set count to prevent hang.
158F 3918 10$:
158F 3919 PUSHL MBA$$_SR(R4) ; Copy MBA status register to stack.
```

0090 C5 52 53 54 C3 1532 3882
52 FC00 C2 9E 1536 3883
0091 C5 52 0080 8F A7 1538 3884
0090 C5 20 85 1543 3885
08 A4 DD 1548 3886
1548 3887
154E 3888
154E 3889
154E 3890
1556 3891
1556 3892
1556 3893
1556 3894
1556 3895
1556 3896
1556 3897
1556 3898
1556 3899
8E 00000100 8F D3 1559 3901
0040 8F A8 1565 3903
68 A5 1569 3904
1568 3905
1568 3906
51 0090 C5 9A 1568 3907
51 01 51 78 1570 3908
10 A3 51 D0 1574 3909
1578 3910
52 FA 8F 9A 1578 3911
FD >2 F5 157C 3912 5\$:
157F 3913
52 54 A5 3C 157F 3914
09 9A 1583 3915
30 A342 1585 3916
52 001E8480 8F D0 1588 3917
08 A4 DD 158F 3918 10\$:
158F 3919

```
8E 00010000 8F D3 1592 3920 BITL #MBASM_SR_ATTEN,(SP)+ ; Wait for SENSE to set MBA attention
      1599 3921 ; bit on.
      03 12 1599 3922 BNEQ 20$ ; NEQ implies the bit came on.
      F1 52 F5 159B 3923 SOBGTR R2,10$ ; Branch back to test again until
      159E 3924 ; count overflows.
      159E 3925 20$:
      10 A3 51 D3 159E 3926 BITL R1,MF_AB(R3) ; Did we get attention yet?
      08 12 15A2 3927 BNEQ 30$ ; NEQ implies that TM78 attention set.
      0810 8F AA 15A4 3928 BICW #UCBSM_ONLINE!UCBSM_VALID,- ; If attention not set yet,
      64 A5 15A8 3929 UCBSW_STS(R5) ; set unit offline/invalid.
      40 11 15AA 3930 BRB 40$ ; And branch around.
      15AC 3931 30$:
      0154 C5 DD 15AC 3932 PUSHL UCBSL_MF_DT(R5) ; Save current value in case at POWERFAIL
      18 A3 D0 15B0 3933 MOVL MF_DT(R3),- ; And move new value to UCB so as to
      0154 C5 15B3 3934 UCBSL_MF_DT(R5) ; subroutine TF_DTYPE.
      10 A8 15B6 3935 BISW #UCBSM_ONLINE,- ; TF DTYPE assumes this bit is set and
      64 A5 15B8 3936 UCBSW_STS(R5) ; clears if it does not like DRIVE TYPE
      00EF 30 15BA 3937 BSBW TF_DTYPE ; CLASSIFY DRIVE TYPE
      0154 C5 8ED0 15BD 3938 POPL UCBSL_MF_DT(R5) ; Restore in case we are in POWERFAIL.
      15C2 3939
      0158 C5 DD 15C2 3940 PUSHL UCBSL_MF_DS(R5) ; Save current value in case at POWERFAIL
      1C A3 D0 15C6 3941 MOVL MF_DS(R3),- ; And move new value to UCB so as to
      0158 C5 15C9 3942 UCBSL_MF_DS(R5) ; call subroutine RECORD_SENSE_INFO
      0118 30 15CC 3943 BSBW RECORD_SENSE_INFO ; Record the sense info in UCB.
      51 0090 C5 9A 15CF 3944 MOVZBL UCBSB_SLAVE(R5),R1 ; R1 = MBA port number for TM78.
      51 01 51 78 15D4 3945 ASHL R1,#1,R1 ; R1 = mask corresponding to attention bit.
      10 A3 51 D0 15D8 3946 MOVL R1,MF_AB(R3) ; Clear attention bit.
      0158 C5 8ED0 15DC 3947 POPL UCBSL_MF_DS(R5) ; Restore in case we are in POWERFAIL.
      15E1 3948
      04 E0 15E1 3949 BBS #UCBSV_ONLINE,- ; If SET, OK drive.
      06 64 A5 15E3 3950 UCBSW_STS(R5),40$
      0800 8F AA 15E6 3951 BICW #UCBSM_VALID,- ; Clear VOLUME SOFTWARE VALID
      64 A5 15EA 3952 UCBSW_STS(R5)
      15EC 3953 40$:
      00B8 C5 B4 15EC 3954 CLRW UCBSW_MF_TMPLTC(R5) ; Clear template tape control register.
      54 A5 F0 15F0 3955 INSV UCBSW_UNIT(R5),- ; Put Unit number of drive
      00 15F3 3956 #MF_TC_V_CMDADR,- ; in template tape control register.
      02 15F4 3957 #MF_TC_S_CMDADR,-
      00B8 C5 15F5 3958 UCBSW_MF_TMPLTC(R5)
      15F8 3959
      01 F0 15F8 3960 INSV #1,- ; Initialize the record count to 1
      02 15FA 3961 #MF_TC_V_RC,- ; in template tape control register.
      06 15FB 3962 #MF_TC_S_RC,-
      00B8 C5 15FC 3963 UCBSW_MF_TMPLTC(R5)
      15FF 3964
      00 F0 15FF 3965 INSV #MFSK_FORMAT_NORMAL_11,-; Initialize format field
      0C 1601 3966 #MF_TC_V_FMT,- ; in template tape control register.
      03 1602 3967 #MF_TC_S_FMT,-
      00B8 C5 1603 3968 UCBSW_MF_TMPLTC(R5)
      0096 8F B0 1606 3969 MOVW #TU78_MAX_REWIND,-
      00BE C5 160A 3970 UCBSW_MF_MAX_REWIND(R5) ; Init UCB field.
      160D 3971
      160D 3972 ;
      160D 3973 ; Here we determine the address of the PTE for the EXTENDED SENSE area
      160D 3974 ; in the UCB so as to be able to efficiently execute this command
      160D 3975 ; during system operation. We also determine this area's offset in
      160D 3976 ; its page. The results of these calculations are deposited in
```

```
160D 3977 : UCB$M_MF_SVAPTE and UCB$M_MF_BOFF respectively. In order to
160D 3978 : effect an EXTENDED SENSE command during system operation, we
160D 3979 : move the contents of these items in the UCB to UCB$M_SVAPTE and
160D 3980 : UCB$M_BOFF, prior to invoking the macros that load the appropriate
160D 3981 : MBA registers.
160D 3982 :
160D 3983 :
160D 3984 :
1612 3985 : MOVAB UCB$M_MF_EXSNS(R5),R0 ; R0 => EXTENDED SENSE area in UCB.
1612 3986 : ASSUME UCB$M_MF_BOFF+2 EQ UCB$M_MF_BCNT
1612 3987 : EXTZV #0,#9,R0,- ; Extract byte offset of area in page.
1619 3988 : UCB$M_MF_BOFF(R5) ; (also clear next word UCB$M_MF_BCNT)
1619 3989 : MOVZBW #60,UCB$M_MF_BCNT(R5) ; Init EXTENDED SENSE byte count.
161E 3990 : EXTZV #9,#21,R0,R0- ; R0 = VPN of area in System Space.
1623 3991 : MOVL G^MMG$GL_SPTBASE,R1 ; R1 => System Page Table base.
162A 3992 : MOVAL (R1)[R0],-
1630 3993 : UCB$M_MF_SVAPTE(R5) ; Point to PTE for area.
1630 3994 :
1630 3995 50$:
1630 3996 : POPL MBA$M_SR(R4) ; CLEAR MBA STATUS
1634 3997 : RSB
```

50 0188 C5 9E

00D4 C5 50 09 00 EF

00D6 C5 3C 9B

50 50 15 09 EF

51 00000000'GF D0

00D0 C5 6140 DE

08 A4 8ED0

05

```
1635 3999 .SBTTL TM78/TU78 UNSOLICITED INTERRUPT PROCESSING
1635 4000 :
1635 4001 : TF_UN SOLNT - TM78/TU78 UNSOLICITED INTERRUPT PROCESSING
1635 4002 :
1635 4003 : This routine is called when an unsolicited interrupt is received for a unit.
1635 4004 : Only three types of unsolicited interrupts (i.e. TM78 initiated) can
1635 4005 : occur. These are DONE interrupts which occur when a REWIND operation
1635 4006 : terminates; ON-LINE interrupts which occur when a drive makes the
1635 4007 : transition to online with a tape volume loaded; and finally hardware
1635 4008 : fault interrupts. The subroutine determines which class of interrupt
1635 4009 : has occurred and branches to code to handle that type of interrupt.
1635 4010 :
1635 4011 : INPUTS:
1635 4012 :
1635 4013 : R3 = Address of TM78 drive control register (CSR)
1635 4014 : R5 = Address of the device unit UCB
1635 4015 : UCB$W_MF_NDTA(R5) = Copy of the Non-data transfer attention register.
1635 4016 :
1635 4017 : OUTPUTS:
1635 4018 : Depending upon the class of interrupt, various bits are set and
1635 4019 : cleared. See below for details of how each class of
1635 4020 : unsolicited interrupt is handled.
1635 4021 :
1635 4022 :
1635 4023 :
1635 4024 :
1635 4025 TF_UN SOLNT: ; UNSOLICITED INTERRUPT PROCESSING
1635 4026 PUSH R2
1637 4027 EXTZV #MF_NDTA_V_NDIC,-
1639 4028 #MF_NDTA_S_NDIC,-
163A 4029 UCB$L_MF_NDTA(R5),R2 ; R2 = Hardware interrupt code.
163E 4030 CMPB #HIC_DONE,R2 ; See if a REWIND is done.
1641 4031 BEQL 30$ ; Branch if DONE.
1643 4032 CMPB #HIC_ON_LINE,R2 ; See if unit has come ON-LINE.
1646 4033 BEQL 50$ ; Branch if so.
1648 4034 :
1648 4035 : If we are here we have experienced some sort of attention for which no one
1648 4036 : is waiting. If so we just ignore it.
1648 4037 :
1648 4038 :
1648 4039 BRB 70$
164A 4040 :
164A 4041 :
164A 4042 : We come here if the unsolicited interrupt was a DONE. This implies that
164A 4043 : a REWIND may have completed.
164A 4044 :
164A 4045 :
164A 4046 30$: CMPB #F_REWIND!GO_BIT,- ; Was last command executed a REWIND
164C 4047 UCB$L_MF_CMD(R5) ; IF EQL YES
164F 4048 BEQL 40$ ; Or was it a DSE (ERASE REST OF TAPE
1651 4049 CMPB #F_DSE!GO_BIT,- ; AND REWIND) command?
1653 4050 UCB$L_MF_CMD(R5) ; If NEQ just noise, branch to end.
1656 4051 BNEQ 70$
1658 4052 40$: BICW #<MTSM_EOF!- ; We know that we are at BOT so update
1658 4053 MTSM_LOST!- ; fields in UCB$L_DEVDEPEND to reflect
1659 4054
```

```

      1659 4056      MTSM EOT>@-16,-      ; our exact position knowledge.
46 A5 16      1659 4057      UCBSL_DEVDEPEND+2(R5)
      01      A8      165C 4058      BISW      #<MTSM BOT@-16>,-      ; Show that we are at BOT.
      46 A5      165E 4059      UCBSL_DEVDEPEND+2(R5)
00C8 C5 00B0 C5 D4      1660 4060      CLRL      UCBSL_RECORD(R5)      ; Show absolute tape position.
      02      CE      1664 4061      MNEGL      #2,UCBSL_MF_PREVTM(R5)      ; Also show ignorance of last TAPEMARK.
      3D      11      1669 4062      BRB      70$      ; Branch around to return.
      166B 4063
      166B 4064      ;
      166B 4065      ; If we come here we have experienced an ON-LINE interrupt; i.e. the unit
      166B 4066      ; has made a transition to ON-LINE with a volume loaded. Also note
      166B 4067      ; that SENSE information is valid at this time.
      166B 4068      ;
      166B 4069      ;
      166B 4070 50$:
0154 C5 18 A3 D0      166B 4071      MOVL      MF_DT(R3),UCBSL_MF_DT(R5)      ; Save SENSE data which
0158 C5 1C A3 D0      1671 4072      MOVL      MF_DS(R3),UCBSL_MF_DS(R5)      ; is valid after an ON-LINE
015C C5 20 A3 D0      1677 4073      MOVL      MF_SN(R3),UCBSL_MF_SN(R5)      ; interrupt.
      0067 3D      167D 4074      BSBW      RECORD_SENSE_INFO      ; Call to update SENSE info to
      1680 4075      ; appropriate UCB fields.
      1680 4076
      10      A8      1680 4077      BISW      #UCBSM_ONLINE,-
      64 A5      1682 4078      UCBSW_STS(R5)      ; Tentatively mark unit ON-LINE.
      0025 30      1684 4079      BSBW      TF_DTYPE      ; Classify drive type.
      1687 4080      ; later ONLINE to OFFLINE transition.
0168 C5 D4      1687 4081      CLRL      UCBSL_MF_NDTA(R5)      ; Prevent a redundant call to TF_UNSOINT
      168B 4082      ; (which might occur if we were waiting
      168B 4083      ; for the UCBSM_MF_REWIND bit to go
      168B 4084      ; zero).
      168B 4085      ;
      168B 4086      ;
      168B 4087      ; Now we either clear, set or leave as is the UCBSM_VALID bit depending on
      168B 4088      ; conditions. If we are here due to a power failure, we leave the
      168B 4089      ; valid bit as is. If the unit has a volume that is mounted FOREIGN,
      168B 4090      ; we set the bit to mark the volume as valid. In all other
      168B 4091      ; circumstances we clear the bit since a manual intervention has
      168B 4092      ; taken place on the drive and we can no longer be sure of the
      168B 4093      ; volume.
      168B 4094      ;
      168B 4095      ;
      18 05      E0      168B 4096      BBS      #UCBSV_POWER,-
      64 A5      168D 4097      UCBSW_STS(R5),70$      ; If powerfail, leave as is and branch.
      13      E1      1690 4098      BBC      #DEV$V_MNT,-
      0D 38 A5      1692 4099      UCBSL_DEVCHAR(R5),60$      ; If NOT mounted, branch to invalidate.
      18      E1      1695 4100      BBC      #DEV$V_FOR,-
      08 38 A5      1697 4101      UCBSL_DEVCHAR(R5),60$      ; If mounted but NOT foreign, go to invalida
      0800 8F A8      169A 4102      BISW      #UCBSM_VALID,-      ; Tape is mounted foreign, so
      64 A5      169E 4103      UCBSW_STS(R5)      ; set volume valid.
      06      11      16A0 4104      BRB      70$      ; Branch around to return.
      0800 8F AA      16A2 4105 60$:      BICW      #UCBSM_VALID,-
      64 A5      16A6 4106      UCBSW_STS(R5)      ; Mark volume software INVALID.
      52 8ED0      16A8 4107 70$:
      05      16AB 4108      POPL      R2
      16AB 4109      RSB      ;

```

```
16AC 4111 .SBTTL TM78 Classify Drive Type and set parameters.
16AC 4112
16AC 4113 : TF_DTYPE - Routine called when an unsolicited ON-LINE interrupt occurs
16AC 4114 : on a drive or at unit initialization time at system startup or
16AC 4115 : after power recovery.
16AC 4116
16AC 4117 : INPUTS:
16AC 4118 : R5 => UCB
16AC 4119 : UCB$W_MF_DT(R5) contains a valid copy of the Drive Type register.
16AC 4120
16AC 4121 : OUTPUTS:
16AC 4122 : UCB$B_DEVTYPE is set to valid value.
16AC 4123 : UCB$M_ONLINE in UCB$W_STS is cleared if the device type does not
16AC 4124 : match any valid device connected to a TM78 controller.
16AC 4125
16AC 4126
16AC 4127 TF_DTYPE:
16AC 4128 BICL3 #^C<MF DT M DTN>,-
16B2 4129 UCB$M_MF_DT(R5),-(SP) ; Drive type to top of stack.
16B6 4130 MOVAB TF_DTDESC,R2 ; R2 => Drive descriptor table.
10$: 4131 CMPW (SP),(R2)+ ; Drive type match?
16BE 4132 BEQL 20$ ; If EQL yes.
16C0 4133 ASSUME TF_DTDESCLEN-2 EQ 1 ; Assumption allows subsequent INCL and DECL
16C0 4134 INCL R2 ; Advance R2 to next entry.
16C2 4135 TSTW (R2) ; End of table?
16C4 4136 BNEQ 10$ ; If NEQ no.
16C6 4137 BICW #UCB$M_ONLINE,-
16C8 4138 UCB$W_STS(R5) ; If unknown type, set unit offline.
16CA 4139 DECL R2 ; Back up to last driver descriptor.
20$: 4140 MOVAB (R2),UCB$B_DEVTYPE(R5) ; Set drive type.
16D0 4141 CLRL UCB$M_MEDIA_ID(R5) ; Assume type unknown
16D4 4142 CMPB #DT$_TU78,UCB$B_DEVTYPE(R5) ; If type not a TU78 then branch
16D9 4143 BNEQ 30$
16DB 4144 MOVL #MEDIA_ID_TU78,UCB$M_MEDIA_ID(R5) ; Set drive media id
16E4 4145 30$: TSTL (SP)+ ; Erase drive type from stack.
16E6 4146 RSB ; Return to caller.
```

FFFFFE00 8F CB 16AC 4128
7E 0154 C5 16B2 4129
52 E97E CF 9E 16B6 4130
82 6E B1 16B8 4131
OC 13 16BE 4132
52 D6 16C0 4133
62 B5 16C0 4134
F5 12 16C2 4135
10 AA 16C4 4136
64 A5 16C6 4137
52 D7 16C8 4138
41 A5 62 90 16CA 4139
008C C5 D4 16CC 4140
41 A5 00 8F 91 16CD 4141
09 12 16D0 4142
008C C5 6D29504E 8F D0 16D9 4143
8E D5 16DB 4144
05 16E4 4145
16E6 4146

```
16E7 4148 .SBTTL RECORD_SENSE_INFO
16E7 4149
16E7 4150 : RECORD_SENSE_INFO - subroutine to update UCB$DEVDEPEND fields given
16E7 4151 : that we currently have valid sense information in hand.
16E7 4152 :
16E7 4153 : INPUTS:
16E7 4154 : R5 => UCB of interest.
16E7 4155 : UCB$MF_DS(R5) is a valid copy of TM78 register.
16E7 4156 :
16E7 4157
16E7 4158 RECORD_SENSE_INFO:
AA 16E7 4159 BICW #<MTSM_BOT!- : We have exact SENSE information as
16E8 4160 MTSM_EOF!- : to position (i.e. BOT and EOT) and
16E8 4161 MTSM_EOT!- : also whether we have a RING or not.
16E8 4162 MTSM_HWL>@-16,- : So we clear these bits and then set
16E8 4163 UCB$DEVDEPEND+2(R5) : those which still hold true.
16E8 4164
16E8 4165 BBC #MF_DS V BOT,- : Branch if NOT at BOT
16ED 4166 UCB$MF_DS(R5),10$
16F1 4167 BISW #<MTSM_BOT@-16>,- : Set BOT bit on in UCB field.
16F3 4168 UCB$DEVDEPEND+2(R5)
16F5 4169 BICW #<MTSM_LOST@-16>,- : Clear the fact that we may have
16F7 4170 UCB$DEVDEPEND+2(R5) : been lost
16F9 4171 CLRL UCB$RECORD(R5) : Set tape position marker to zero.
16FD 4172 MNEGL #2,UCB$MF_PREVTM(R5) : And show ignorance of TAPEMARKS.
1702 4173 10$:
1702 4174 BBC #MF_DS V EOT,- : Branch if not beyond EOT marker.
1704 4175 UCB$MF_DS(R5),20$
1708 4176 BISW #<MTSM_EOT@-16>,- : Set bit if we are indeed beyond the
170A 4177 UCB$DEVDEPEND+2(R5) : EOT marker.
170C 4178 20$:
170C 4179 BBC #MF_DS V FPT,- : Branch around if NO ring present on
170E 4180 UCB$MF_DS(R5),30$ : tape reel.
1712 4181 BISW #<MTSM_HQL@-16>,- : Set bit if NO ring present on
1714 4182 UCB$DEVDEPEND+2(R5) : tape reel.
1716 4183 30$:
1716 4184 BBS #MTSV_LOST,- : Branch around if LOST.
1718 4185 UCB$DEVDEPEND(R5),40$
171B 4186 CMPL UCB$RECORD(R5),- : See if we are at a TAPEMARK by comparing
171F 4187 UCB$MF_PREVTM(R5) : current position with last TAPEMARK.
1722 4188 BNEQ 40$ : NEQ implies not at a TAPEMARK.
1724 4189 BISW #<MTSM_EOF@-16>,- : Set bit indicating currently positioned
1726 4190 UCB$DEVDEPEND+2(R5) : at a TAPEMARK.
1728 4191 40$:
1728 4192 BBS #MF_DS V BOT,- : If at BOT, then ignore what drive
172A 4193 UCB$MF_DS(R5),60$ : and leave UCB density as is.
172E 4194 ASSUME MF$K_DENSITY 1600 EQ 0 : Here we find out the drive density
172E 4195 CLR B UCB$B_MF_DENSITY(R5) : and set UCB variables accordingly.
1732 4196 PUSH R0 : We set the density to the default 1600
1734 4197 MOVL #MT$K_PE_1600,R0 : and prepare to test the actual status
1737 4198 BBS #MF_DS V PE,- : Branch if 1600 BPI bit set in device
1739 4199 UCB$MF_DS(R5),50$ : status register.
173D 4200 ASSUME MF$K_DENSITY 6250 EQ 1 : We fall thru here if the drive is 6250
173D 4201 INC B UCB$B_MF_DENSITY(R5) : We set density to 6250 in that case
1741 4202 MOVL #MT$K_GCR_6250,R0 : And get set to update UCB$DEVDEPEND
1744 4203 50$:
1744 4204 INSV R0,- : Insert the density value into the
```

- TM78/TU78 MAGTAPE DRIVER
RECORD_SENSE_INFO

```
16-SEP-1984 00:08:15 VAX/VMS Macro V04-00
5-SEP-1984 00:17:37 [DRIVER.SRC]TFDRIVER.MAR;1
```

Page 93
(1)

			1746	4205	
05	08		1746	4206	
44	A5		1748	4207	
	50	8ED0	174A	4208	
			174D	4209	60\$:
	05		174D	4210	POPL
					RSB

```

#MTSV_DENSITY,-      ; proper field of UCBSL_DEVDEPEND
#MTSS_DENSITY,-      ; so as to indicate the current
UCBSL_DEVDEPEND(R5)   ; density passed in R0.
R0                    ; Restore

; Return to caller.

```

[illegible]


```
174E 4212 : SAVE_TM78_REGS - Subroutine to simply copy all TM78 registers to the UCB.
174E 4213 :
174E 4214 :
174E 4215 : INPUTS:
174E 4216 :
174E 4217 :     R3 => TM78 CSR
174E 4218 :     R5 => UCB
174E 4219 :
174E 4220 : OUTPUTS:
174E 4221 :
174E 4222 :     Register values saved in the UCB.
174E 4223 :
174E 4224 : Note that since this routine is called at interrupt time (i.e. between
174E 4225 : WFIKPCB and IOFORK) it must conform to the register conventions in
174E 4226 : force at that time, which include NOT destroying registers R0 and R1.
174E 4227 :
174E 4228 :
174E 4229 : SAVE_TM78_REGS:
174E 4230 :
174E 4231 :     ASSUME UCB$$_MF_IS-UCB$$_MF_CS1 EQ MF_IS
174E 4232 :     ASSUME UCB$$_MF_TC-UCB$$_MF_CS1 EQ MF_TC
174E 4233 :     ASSUME UCB$$_MF_MR1-UCB$$_MF_CS1 EQ MF_MR1
174E 4234 :     ASSUME UCB$$_MF_AB-UCB$$_MF_CS1 EQ MF_AB
174E 4235 :     ASSUME UCB$$_MF_BC-UCB$$_MF_CS1 EQ MF_BC
174E 4236 :     ASSUME UCB$$_MF_DT-UCB$$_MF_CS1 EQ MF_DT
174E 4237 :     ASSUME UCB$$_MF_DS-UCB$$_MF_CS1 EQ MF_DS
174E 4238 :     ASSUME UCB$$_MF_SN-UCB$$_MF_CS1 EQ MF_SN
174E 4239 :     ASSUME UCB$$_MF_MR2-UCB$$_MF_CS1 EQ MF_MR2
174E 4240 :     ASSUME UCB$$_MF_MR3-UCB$$_MF_CS1 EQ MF_MR3
174E 4241 :     ASSUME UCB$$_MF_NDTA-UCB$$_MF_CS1 EQ MF_NDTA
174E 4242 :     ASSUME UCB$$_MF_NDT0-UCB$$_MF_CS1 EQ MF_NDT0
174E 4243 :     ASSUME UCB$$_MF_NDT1-UCB$$_MF_CS1 EQ MF_NDT1
174E 4244 :     ASSUME UCB$$_MF_NDT2-UCB$$_MF_CS1 EQ MF_NDT2
174E 4245 :     ASSUME UCB$$_MF_NDT3-UCB$$_MF_CS1 EQ MF_NDT3
174E 4246 :
174E 4247 :     PUSHF #M<R0,R1,R2> : Save registers.
174E 4248 :     MOVAB UCB$$_MF_CS1(R5),R0 : R0 => Block in UCB where we save regs
174E 4249 :     MOVAB MF_CST(R3),R1 : R1 => start of TM78 registers.
174E 4250 :     MOVL #MF_NDT3+4-MF_CS1/4,R2 : R2 = number of registers to copy.
174E 4251 :
174E 4252 :     MOVL (R1)+,(R0)+ : Copy one Register to UCB.
174E 4253 :     SOBGTR R2,10$ : Loop thru all contiguous registers.
174E 4254 :     MOVL MF_ID(R3),- :
174E 4255 :     UCB$$_MF_ID(R5) : Copy non-contiguous register.
174E 4256 :     POPR #M<R0,R1,R2> : Restore registers.
174E 4257 :     RSB : Return to caller.
```

50 013C 07 BB 174E 4247 : PUSHF #M<R0,R1,R2> : Save registers.
51 63 9E 1750 4248 : MOVAB UCB\$\$_MF_CS1(R5),R0 : R0 => Block in UCB where we save regs
52 10 D0 1755 4249 : MOVAB MF_CST(R3),R1 : R1 => start of TM78 registers.
80 81 D0 1758 4250 : MOVL #MF_NDT3+4-MF_CS1/4,R2 : R2 = number of registers to copy.
FA 52 F5 175B 4251 :
44 A3 D0 175B 4252 : MOVL (R1)+,(R0)+ : Copy one Register to UCB.
017C C5 175E 4253 : SOBGTR R2,10\$: Loop thru all contiguous registers.
07 BA 1761 4254 : MOVL MF_ID(R3),- :
05 1764 4255 : UCB\$\$_MF_ID(R5) : Copy non-contiguous register.
1767 4256 : POPR #M<R0,R1,R2> : Restore registers.
1769 4257 : RSB : Return to caller.

```
176A 4259 : SAVE_MBA_REGS - Routine to record non-volatile MBA registers involved
176A 4260 : in Data Transfer operations for ERROR LOGGING use.
176A 4261 :
176A 4262 : INPUTS:
176A 4263 :
176A 4264 :     R3 => TM78 CSR
176A 4265 :     R4 => MBA CSR
176A 4266 :     R5 => UCB
176A 4267 :
176A 4268 : OUTPUTS:
176A 4269 :
176A 4270 :     Register values saved in the UCB.
176A 4271 :
176A 4272 :
176A 4273 : Note that since this routine is called at interrupt time (i.e. between
176A 4274 : WFIKPC and IOFORK) it must conform to the register conventions in
176A 4275 : force at that time, which include NOT destroying registers R0 and R1.
176A 4276 :
176A 4277 :
176A 4278 : SAVE_MBA_REGS:
176A 4279 :
176A 4280 :     ASSUME UCB$MFMBA_CSR+4 EQ UCB$MFMBA_CR
176A 4281 :     ASSUME UCB$MFMBA_CR+4 EQ UCB$MFMBA_SR
176A 4282 :     ASSUME UCB$MFMBA_SR+4 EQ UCB$MFMBA_VAR
176A 4283 :     ASSUME UCB$MFMBA_VAR+4 EQ UCB$MFMBA_BCR
176A 4284 :     ASSUME UCB$MFMBA_BCR+4 EQ UCB$MFMBA_FMAP
176A 4285 :     ASSUME UCB$MFMBA_FMAP+4 EQ UCB$MFMBA_PMAP
176A 4286 :
176A 4287 :     PUSH R0,R1 ; Save registers.
176C 4288 :     MOVAB UCB$MFMBA_CSR(R5),R1 ; R1 => MBA dump area in UCB.
1771 4289 :     MOVL MBA$CSR(R4),(R1)+ ; Save MBA CSR contents
1774 4290 :     MOVL MBA$CR(R4),(R1)+ ; Save MBA Control Reg.
1778 4291 :     MOVL MBA$SR(R4),(R1)+ ; Save MBA Status Reg.
177C 4292 :     MOVL MBA$VAR(R4),(R1)+ ; Save MBA Virtual ADDR Reg.
1780 4293 :     MOVL MBA$BCR(R4),(R1)+ ; Save MBA Byte Count Reg.
1784 4294 :     EXTZV #9,#8,-8(R1),R0 ; R0 = Final Map Reg. Number.
178A 4295 :     ; NOTE: -8(R1) corresponds to
178A 4296 :     ; UCB$MFMBA_VAR(R5).
178A 4297 :     MOVL MBA$MAP(R4)[R0],(R1)+ ; Save Final Map Register.
1790 4298 :     CLRL (R1) ; Zero UCB$MFMBA_PMAP in case
1792 4299 :     ; there is NO previous Map Reg.
1792 4300 :     DECL R0 ; R0 = Previous Map Reg #
1794 4301 :     BLSS 10$ ; LSS implies NO previous.
1796 4302 :     MOVL MBA$MAP(R4)[R0],(R1) ; Save Previous Map Register.
179C 4303 :
179C 4304 :
179E 4305 :     POP R0,R1 ; Restore registers.
179E 4305 :     RSB
```

50 F8 A1 08 09 EF 1784 4294 EXTZV #9,#8,-8(R1),R0

81 0800 C440 D0 178A 4297 MOVL MBA\$MAP(R4)[R0],(R1)+

61 0800 C440 D0 1796 4302 MOVL MBA\$MAP(R4)[R0],(R1)

03 BA 179C 4304 POP R0,R1

05 179E 4305 RSB

```
179F 4307 .SBTTL TM78_NOTREADY (PRIOR and POST)
179F 4308
179F 4309 : Code branched to when either of the bits UCBSM_MF_TMRDY or UCBSM_MF_ATTN
179F 4310 : is found set to 1, either PRIOR to losing control via a WFIKPCB
179F 4311 : or just after regaining control after a WFIKPCB (POST). The
179F 4312 : setting of either bit means the TM78 is out of whack and must be
179F 4313 : reset. If the UCBSM_MF_ATTN bit is set then we must reset the TM78.
179F 4314 : If UCBSM_MF_ATTN is not set but the other one is we must wait for
179F 4315 : the TM78 to come ready again after it is reset. The first entry here
179F 4316 : is branched to when one of the bits is found on PRIOR to the WFIKPCB
179F 4317 : and therefore we have just issued a DSBINT macro. Thus we must
179F 4318 : perform an ENBINT to undo this. The second entry is branched to
179F 4319 : after a WFIKPCB so we must IOFORK to get down from interrupt state.
179F 4320
179F 4321 : INPUTS:
179F 4322 : R3 => TM78 CSR.
179F 4323 : R5 => UCB
179F 4324 :
179F 4325
179F 4326 .ENABLE LSB
179F 4327 TM78_NOTREADY PRIOR:
179F 4328 ENBINT
06 11 17A2 4329 BRB 10$
17A4 4330 TM78_NOTREADY POST:
17A4 4331 IOFORK
17AA 4332 10$:
09 05 E1 17AA 4333 BBC #UCBSM_MF_ATTN - ; Is it our responsibility to TMCLR?
68 A5 17AC 4334 UCBSM_DEVSTS(R5),20$ ; Bit clear implies NO, so branch.
00BF 30 17AF 4335 BSBW ISSUE-TMCLR
03 50 E8 17B2 4336 BLBS R0,20$ ; LBS implies successful TMCLR.
FBDA 31 17B5 4337 BRW FUNCTION_EXIT ; If failure, exit.
17B8 4338 20$:
17B8 4339 RELCHAN ; In case we own channel release it.
03 04 E1 17BE 4340 BBC #UCBSM_MF_TMRDY - ; See if we must await the TM78.
68 A5 17C0 4341 UCBSM_DEVSTS(R5),30$ ; Clear bit means NO, so branch.
0065 30 17C3 4342 BSBW AWAIT-TMRDY
17C6 4343 30$:
08 A8 17C6 4344 BISW #UCBSM_MF_REPOS - ; Signal need to reposition.
68 A5 17C8 4345 BSBW UCBSM_DEVSTS(R5)
F6A9 30 17CA 4346 BSBW REPOSITION ; Call but if successful it branches
17CD 4347 ; to TF_RESTARTIO.
08 AA 17CD 4348 BICW #UCBSM_MF_REPOS - ; Clear bit if unsuccessful reposition
68 A5 17CF 4349 UCBSM_DEVSTS(R5) ; operation.
50 0054 8F 3C 17D1 4350 MOVZWL #SSS_CTRLERR,R0 ; Indicate final I/O status.
FBB9 31 17D6 4351 BRW FUNCTION_EXIT
17D9 4352 .DISABLE CSB
```

```
17D9 4354 .SBTTL FAULT_INTERRUPT
17D9 4355
17D9 4356 : FAULT_INTERRUPT - subroutine called at interrupt level when a 'B' type
17D9 4357 : fault interrupt has been received. The strategy here is as follows:
17D9 4358 :
17D9 4359 : 1. We set the UCBSM_MF_ATTN bit in our UCBSW_DEVSTS to indicate that
17D9 4360 : it is our responsibility to do the TMCLR.
17D9 4361 : 2. We loop thru all UCBs associated with this TM78 and set the
17D9 4362 : UCBSM_MF_TMRDY bits in each UCBSW_DEVSTS to indicate to
17D9 4363 : processes that may currently be using these UCBs that they
17D9 4364 : must WAIT TMRDY prior to proceeding with use of the TM78.
17D9 4365 : 3. We next determine whether we ISSUE_TMCLR inline (via IOFORK)
17D9 4366 : or simply return to our caller confident that we will shortly
17D9 4367 : call ISSUE_TMCLR at fork level. This determination is made
17D9 4368 : on the following basis: if our UCB is NOT busy (i.e.
17D9 4369 : UCBSM_BSY is zero in UCBSW_STS) then we do the ISSUE_TMCLR
17D9 4370 : inline. Also if the UCB IS busy but bit UCBSM_MF_UCBFREE
17D9 4371 : in our UCBSW_DEVSTS is set then we can call ISSUE_TMCLR
17D9 4372 : inline. In all other circumstances (i.e. BUSY and
17D9 4373 : UCBSM_MF_UCBFREE zero) we simply RSB to the interrupt
17D9 4374 : handler.
17D9 4375 : 4. If we decide to proceed inline, we set bit UCBSM_MF_UCBBUSY
17D9 4376 : in our UCBSW_DEVSTS to signal the fact that we are going
17D9 4377 : to use the UCB fork block asynchronously and that anyone
17D9 4378 : entering STARTIO had better defer until we finish.
17D9 4379 : 5. We IOFORK to get out of interrupt context and then we call
17D9 4380 : ISSUE_TMCLR.
17D9 4381 : 6. We reset UCBSM_MF_UCBBUSY.
17D9 4382 : 7. Now we either evaporate (i.e. RSB) or we go to startup a pending
17D9 4383 : I/O operation. The determining factor in this is whether
17D9 4384 : the UCBSM_BSY bit is set in UCBSW_STS. A set bit means that
17D9 4385 : some process has entered STARTIO since we began inline
17D9 4386 : operation and therefore deferred its operation. It therefore
17D9 4387 : is our responsibility to start this operation. To do it
17D9 4388 : we clear the UCBSM_MF_RSTCNT to indicate that we are starting
17D9 4389 : an operation and then to branch to TF_RESTARTIO. If the
17D9 4390 : UCBSM_BSY bit was zero we simply RSB.
17D9 4391 :
17D9 4392 : INPUTS:
17D9 4393 : R2 = hardware fault code.
17D9 4394 : R3 => TM78 CSR
17D9 4395 : R4 => TM78 IDB
17D9 4396 : R5 => our UCB.
17D9 4397 :
17D9 4398 :
17D9 4399 : FAULT_INTERRUPT:
17D9 4400 :
17D9 4401 : BISW #UCBSM_MF_ATTN, - ; We must do the ISSUE_TMCLR
17D9 4402 : UCBSW_DEVSTS(R5)
17D9 4403 : PUSHB #^M<R0,R1>
17D9 4404 : CLRL R0 ; Clear loop variable.
17D9 4405 : 10$:
17D9 4406 : MOVL IDBSL_UCBLST(R4)[R0],R1 ; R1 => a UCB.
17D9 4407 : BEQL 20$ ; EQL implies no such UCB.
17D9 4408 : BISW #UCBSM_MF_TMRDY, - ; Tell all others to wait.
17D9 4409 : UCBSW_DEVSTS(R1)
17D9 4410 : 20$: AOBLEQ #3,R0,10$ ; Loop thru all possible UCBs.
```

20 A8 17D9 4401 BISW #UCBSM_MF_ATTN, - ; We must do the ISSUE_TMCLR
60 A5 17D9 4402 UCBSW_DEVSTS(R5)
03 BB 17D9 4403 PUSHB #^M<R0,R1>
50 D4 17D9 4404 CLRL R0 ; Clear loop variable.
51 18 A440 D0 17E1 4405 10\$:
04 13 17E6 4406 MOVL IDBSL_UCBLST(R4)[R0],R1 ; R1 => a UCB.
10 A8 17E8 4407 BEQL 20\$; EQL implies no such UCB.
68 A1 17EA 4408 BISW #UCBSM_MF_TMRDY, - ; Tell all others to wait.
F1 50 03 F3 17EC 4409 UCBSW_DEVSTS(R1)
17EC 4410 20\$: AOBLEQ #3,R0,10\$; Loop thru all possible UCBs.

03	BA	17F0	4411	POPR	#^M<R0,R1>	
		17F2	4412			
09 64	A5	17F2	4413	BBC	#UCBSV BSY,-	
	06	17F4	4414		UCBSW_STS(R5),30\$	; If clear we can go inline, so branch.
04 68	A5	17F7	4415	BBS	#UCBSV MF UCBFREE,-	; If set, UCB busy but FORK block free
	FF4F	17F9	4416		UCBSW_DEVSTS(R5),30\$	; so we branch to continue inline.
		17FC	4417	BSBW	SAVE_TM78_REGS	; Record TM78 register values.
		17FF	4418			
	05	17FF	4419	RSB		; We cannot continue inline, so return.
		1800	4420			
	07	1800	4421	BBSS	#UCBSV MF UCBBUSY,-	; Indicate we are about to asynchronously
FA 68	A5	1802	4422		UCBSW_DEVSTS(R5),25\$	; use the UCB fork block unless it is
		1805	4423			; already in use.
	FF46	1805	4424	BSBW	SAVE_TM78_REGS	; Record TM78 register values.
		1808	4425	IOFORK		; Leave interrupt level.
		180E	4426			
00E4	C5	180E	4427	MOVAB	UCBSL_MF_SUBSTACK(R5),-	; Initialize SUBSTACK pointer in case
00E0	C5	1812	4428		UCBSL_MF_SUBSP(R5)	; it is NG.
	0059	1815	4429	BSBW	ISSUE-TMCLR	
0080	8F	1818	4430	BICW	#UCBSM MF UCBBUSY,-	; We no longer need the UCB fork block.
	68	181C	4431		UCBSW_DEVSTS(R5)	
		181E	4432			
	08	181E	4433	BBC	#UCBSV BSY,-	; If UCB NOT busy, branch.
07 64	A5	1820	4434		UCBSW_STS(R5),40\$	
		1823	4435			
	00BB	1823	4436	CLRB	UCBSB MF_RSTCNT(R5)	; Begin operation with zero count.
	E908	1827	4437	BRW	TF_RESTARTIO	; Go begin I/O operation/
		182A	4438			
	05	182A	4439	RSB		; Evaporate.

```
182B 4441 .SBTTL AWAIT_TMRDY
182B 4442
182B 4443 : AWAIT_TMRDY - subroutine called when the driver finds UCBSM_MF_TMRDY bit
182B 4444 : set in UCBSW_DEVSTS. This bit set means that a 'B' type-fault has
182B 4445 : occurred on some other TU78 drive connected to this TM78 and that
182B 4446 : a TMCLR has been or soon will be issued within the context of the
182B 4447 : UCB associated with the drive that experienced the fault. When the
182B 4448 : TMRDY bit in TM78 register MF_ID comes on after the TMCLR has been
182B 4449 : issued, this other process will reset our UCBSM_MF_TMRDY and we will
182B 4450 : be able to continue. Here we simply wait for this bit to clear. The
182B 4451 : method of waiting is to simply IOFORK. This puts us on the end of
182B 4452 : the fork list. We are guaranteed that the other process which will
182B 4453 : reset our bit is also at fork level and is also looping, via IOFORK,
182B 4454 : awaiting the TM78 hardware bit to come on.
182B 4455
182B 4456 : INPUTS:
182B 4457 : R5 => UCB
182B 4458
182B 4459 : OUTPUTS:
182B 4460 : NONE
182B 4461
182B 4462
182B 4463 AWAIT_TMRDY:
182B 4464 SUBSAVE
1835 4465 SUBPUSH R3
183F 4466 MOVZBL #255,R3 ; Prevent infinite loop.
1843 4467 10$:
1843 4468 BBC #UCBSV_MF_TMRDY,- ; If clear then our wait is over so
1845 4469 UCBSW_DEVSTS(R5),20$ ; branch around.
1848 4470 IOFORK ; Allow other process in to reset bit.
184E 4471 SOBGTR R3,10$ ; Loop a limited number of times.
1851 4472 BICW #UCBSM_MF_TMRDY,- ; If it doesn't get reset, then we do it
1853 4473 UCBSW_DEVSTS(R5) ; ourselves.
1855 4474 MOVZWL #$$$_CTRLERR,R0 ; If it doesn't reset soon, forget it.
185A 4475 BRW FUNCTION_EXIT ; Set error status and goto exit.
185D 4476 20$:
185D 4477 SUBPOP R3
1867 4478 SUBRETURN
```

53 FF 8F 9A
15 68 A5 E1
F2 53 F5
10 AA
68 A5
50 0054 8F 3C
FB35 31

```
1871 4480 .SBTTL ISSUE_TMCLR
1871 4481
1871 4482 : ISSUE_TMCLR - subroutine to invoke extended sense, errorlog, and to
1871 4483 : issue a TMCLR to reset the TM78 hardware which is apparently
1871 4484 : out of whack. After a successful TMCLR we reset the UCBSM_MF_TMRDY
1871 4485 : bits in UCBSW_DEVSTS in all UCBs attached to the TM78.
1871 4486
1871 4487 : INPUTS:
1871 4488 : R5 => UCB
1871 4489
1871 4490 : OUTPUTS:
1871 4491 : R0 = status return.
1871 4492 : $$$NORMAL implies successful TMCLR.
1871 4493 : $$$CTRLERR implies unable to complete TMCLR.
1871 4494
1871 4495 : Registers R1 and R2 are modified due to IOFORKING.
1871 4496
1871 4497
1871 4498 ISSUE_TMCLR:
1871 4499
1871 4500 SUBSAVE
1871 4501 SUBPUSH R3,R4
1871 4502 MOVL UCBSL_CRB(R5),R3 ; R3 => TM78 CRB
1871 4503
1871 4504 : Here make sure that we do not bog down the system with multiple TMCLRs
1871 4505 : to a sick TM78. We do this by refusing to do a TMCLR if less than
1871 4506 : 15 seconds have occurred since the last one. The time of the last
1871 4507 : one is kept in CRBSL_AUXSTRUC.
1871 4508
1871 4509 ADDL3 #15,CRBSL_AUXSTRUC(R3),R0 ; R0 has time 15 seconds after
1871 4510 CMPL R0,G^EXESGL_ABSTIM ; last TMCLR. Is that in the
1871 4511 BGEQU 15$ ; past?. GEQU means no.
1871 4512 MOVL G^EXESGL_ABSTIM,- ; If we are going to do TMCLR,
1871 4513 CRBSL_AUXSTRUC(R3) ; then record time now in CRB.
1871 4514
1871 4515 ASSUME IDBSL_CSR EQ 0
1871 4516 MOVL @CRBSL_INTD+VEC$L_IDB(R3),R3 ; R3 => TM78 CSR
1871 4517
1871 4518 BICW #UCBSM_MF_ATTN!- ; Reset bit that provoked this call.
1871 4519 UCBSM_MF_EXSNS_DONE,- ; and bit that prevents redundant
1871 4520 UCBSW_DEVSTS(R5) ; extended sense operations.
1871 4521 RELCHAN ; Just in case we had channels.
1871 4522
1871 4523 BISL #MF_ID_M_TMCLR,MF_ID(R3); Reset TM78 hardware.
1871 4524 MOVZBL #255,R4 ; Setup count to prevent infinite loop.
1871 4525 10$:
1871 4526 IOFORK ; Waste a little time.
1871 4527 BITL #MF_ID_M_TMRDY,MF_ID(R3); Has TM78 settled down?
1871 4528 BNEQ 20$ ; NEQ implies TM78 now reset.
1871 4529 SOBGTR R4,10$ ; Loop a limited number of times.
1871 4530 15$:
1871 4531 BICW #UCBSM_MF_TMRDY,- ; Either failed or refused to do TMCLR.
1871 4532 UCBSW_DEVSTS(R5) ; In the case of failure, reset our
1871 4533 $$$CTRLERR,R0 ; bit.
1871 4534 BRB 50$ ; Set final I/O status.
1871 4535 20$:
1871 4536 MOVL UCBSL_CRB(R5),R4 ; R4 => TM78 CRB.
```

53 24 A5 D0 188F 4502 MOVL UCBSL_CRB(R5),R3 ; R3 => TM78 CRB

50 10 A3 0F C1 1893 4509 ADDL3 #15,CRBSL_AUXSTRUC(R3),R0 ; R0 has time 15 seconds after
00000000'GF 50 D1 1898 4510 CMPL R0,G^EXESGL_ABSTIM ; last TMCLR. Is that in the
37 1E 189F 4511 BGEQU 15\$; past?. GEQU means no.
00000000'GF D0 18A1 4512 MOVL G^EXESGL_ABSTIM,- ; If we are going to do TMCLR,
10 A3 18A7 4513 CRBSL_AUXSTRUC(R3) ; then record time now in CRB.

53 2C B3 D0 18A9 4516 MOVL @CRBSL_INTD+VEC\$L_IDB(R3),R3 ; R3 => TM78 CSR

68 A5 0220 8F AA 18AD 4518 BICW #UCBSM_MF_ATTN!- ; Reset bit that provoked this call.
18AE 4519 UCBSM_MF_EXSNS_DONE,- ; and bit that prevents redundant
18AE 4520 UCBSW_DEVSTS(R5) ; extended sense operations.

44 A3 00004000 8F C8 18B9 4523 BISL #MF_ID_M_TMCLR,MF_ID(R3); Reset TM78 hardware.
54 FF 8F 9A 18C1 4524 MOVZBL #255,R4 ; Setup count to prevent infinite loop.

44 A3 00008000 8F D3 18CB 4527 BITL #MF_ID_M_TMRDY,MF_ID(R3); Has TM78 settled down?
OE 12 18D3 4528 BNEQ 20\$; NEQ implies TM78 now reset.
ED 54 F5 18D5 4529 SOBGTR R4,10\$; Loop a limited number of times.

50 68 A5 10 AA 18D8 4531 BICW #UCBSM_MF_TMRDY,- ; Either failed or refused to do TMCLR.
0054 8F 3C 18DC 4532 UCBSW_DEVSTS(R5) ; In the case of failure, reset our
1C 11 18E1 4533 \$\$\$CTRLERR,R0 ; bit.
54 24 A5 D0 18E3 4535 BRB 50\$; Set final I/O status.

```
54 2C A4 D0 18E7 4537      MOVL CRBSL_INTD+VECSL_IDB(R4),R4      ; R4 => TM78 IDB.
      50 D4 18EB 4538      CLRL R0                          ; Loop counter.
      18ED 4539 30$:
53 18 A440 D0 18ED 4540      MOVL IDBSL_UCBLST(R4)[R0],R3      ; R3 => a UCB.
      04 13 18F2 4541      BEQL 40$                          ; EQL implies no such UCB.
      10 AA 18F4 4542      BICW #UCBSM_MF TMRDY,-             ; Signal other processes that they
      68 A3 18F6 4543      UCB$W_DEVSTS(R3)                   ; may continue.
F1 50 03 F3 18F8 4544 40$: AOBLEQ #3,R0,30$                  ; Loop thru all 4 possible drives.
      18FC 4545
      50 01 3C 18FC 4546      MOVZWL S^#SS$_NORMAL,R0          ; Set final status.
      18FF 4547 50$:
      18FF 4548      SUBPUSH R0                                ; Save STATUS.
      03 50 E9 1909 4549      BLBC R0,60$                     ; If TMCLR was UNsuccessful, branch,
      190C 4550      ; since if we couldn't TMCLR, we can't
      190C 4551      ; do an EXTENDED SENSE either.
      190C 4552
      F927 30 190C 4553      BSBW INVOKE_EXTENDED_SENSE
      190F 4554 60$:
      0200 8F AA 190F 4555      BICW #UCBSM_MF EXSNS_DONE,-    ; Clear again so we can get SENSE on
      68 A5 1913 4556      UCB$W_DEVSTS(R5)                   ; regular I/O operation errorlog.
      1915 4557
      00000000'GF 16 1915 4558      JSB G^ERL$DEVICEATTN        ; Call special errorlog entry.
      191B 4559
      191B 4560      SUBPOP R0                                  ; Restore STATUS.
      1925 4561      SUBPOP R4,R3
      1939 4562      SUBRETURN
```



```
1943 4564 .SBTTL TM78/TU78 SLAVE CONTROLLER INTERRUPT DISPATCHER
1943 4565 :+
1943 4566 : TFSINT - TM78/TU78 SLAVE CONTROLLER INTERRUPT DISPATCHER
1943 4567 :
1943 4568 : THIS ROUTINE IS ENTERED VIA A JSB INSTRUCTION WHEN AN ATTENTION
1943 4569 : INTERRUPT OCCURS ON A TM78/TU78 SLAVE CONTROLLER.
1943 4570 : THE STATE OF THE STACK ON ENTRY IS:
1943 4571 :
1943 4572 : 00(SP) = ADDRESS OF IDB ADDRESS.
1943 4573 : 04(SP) = SAVED R2.
1943 4574 : 08(SP) = SAVED R3.
1943 4575 : 12(SP) = SAVED R4.
1943 4576 : 16(SP) = SAVED R5.
1943 4577 : 20(SP) = INTERRUPT PC.
1943 4578 : 24(SP) = INTERRUPT PSL.
1943 4579 :
1943 4580 : INTERRUPT DISPATCHING OCCURS AS FOLLOWS:
1943 4581 :
1943 4582 : For DATA TRANSFER interrupts we simply restore registers R3 and R4
1943 4583 : call back the DRIVER at the return address. We determine that we
1943 4584 : are processing a DATA TRANSFER interrupt if both the PRIMARY and
1943 4585 : SECONDARY channels (i.e. the TM78 CRB-IDB and the MBA CRB-IDB) are
1943 4586 : owned by the UCB of interest and the UCB is expecting an interrupt.
1943 4587 :
1943 4588 : If we did NOT have a DATA TRANSFER interrupt we simply branch around
1943 4589 : the call back to the DRIVER.
1943 4590 :
1943 4591 : After return from the DRIVER call back or after branching around the
1943 4592 : DRIVER call back, as the case may be, we proceed to determine if an
1943 4593 : ATTENTION interrupt is present and if so we process.
1943 4594 :
1943 4595 : We determine if the attention bit is set for the TM78. If this is
1943 4596 : not so, we simply branch around to dismiss the interrupt. If
1943 4597 : however the attention bit for the TM78 is set we proceed to read the
1943 4598 : Non-Data Transfer Attention register in order to determine which
1943 4599 : drive caused the ATTENTION. If the unit is expecting an interrupt
1943 4600 : then the DRIVER is called at its interrupt return address. If no
1943 4601 : interrupt was expected, then the DRIVER is called at its UNSOLICITED
1943 4602 : interrupt address. Upon return from either call the attention bit is
1943 4603 : cleared and the interrupt is dismissed.
1943 4604 :
1943 4605 :-
1943 4606 :
1943 4607 TFSINT:: : TM78/TU78 SLAVE CONTROLLER INTERRUPT DISPAT
1943 4608 : MOVL @ (SP), R4 : R4 => IDB.
1943 4609 : MOVL IDB$$_OWNER(R4), R5 : R5 => owner UCB.
1943 4610 : BEQL 10$ : EQL => no owner.
1943 4611 :
1943 4612 : If here we own PRIMARY channel.
1943 4613 : MOVL UCB$$_CRB(R5), R4 : R4 => TM78 CRB
1943 4614 : MOVL CRB$$_LINK(R4), R4 : R4 => MBA CRB
1943 4615 : MOVL CRB$$_INTD+VEC$$_IDB(R4), R4 : R4 => MBA IDB
1943 4616 : CMPL R5, IDB$$_OWNER(R4) : Are we owners of MBA also?
1943 4617 : BNEQ 10$ : If NOT, branch around call back.
1943 4618 :
1943 4619 : If here we also own SECONDARY channel.
1943 4620 : BBCC #UCB$$_INT, - : If clear DATA TRANSFER interrupt not
```

54 00 BE D0 1943 4608
55 04 A4 D0 1947 4609
1E 13 194B 4610
194D 4611
194D 4612
54 24 A5 D0 194D 4613
54 20 A4 D0 1951 4614
54 2C A4 D0 1955 4615
04 A4 55 D1 1959 4616
OC 12 195D 4617
195F 4618
195F 4619
01 E5 195F 4620

```

07 64 A5      1961 4621      UCB$W_STS(R5),10$      ; expected, so branch around call back.
53 10 A5      1964 4622      MOVQ      UCB$L_FR3(R5),R3      ; Restore driver registers R3 and R4.
   0C B5      1968 4623      JSB       @UCB$L_FPC(R5)      ; Call back driver at return address.
   196B 4624
   196B 4625
   196B 4626      ; Here we determine if the TM78 has a pending attention interrupt. To do that
   196B 4627      ; we first determine which port on the MASSBUS is occupied by this TM78.
   196B 4628      ; To do this we obtain a pointer to the MBA-CSR indirectly
   196B 4629      ; through the IDB=>ADP. With the TM78-CSR address already in hand,
   196B 4630      ; we subtract to determine the distance of the TM78-CSR to the base
   196B 4631      ; of the MBA external registers. This difference divided by 128
   196B 4632      ; (right shifted by 7) gives the port number of interest. We then test
   196B 4633      ; the attention bit corresponding to the port number.
   196B 4634
   196B 4635
   54 00 BE      DO 196B 4636 10$:      MOVL      @ (SP),R4      ; Again R4 => IDB.
   53 64 DO      196F 4637      MOVL      IDB$L_CSR(R4),R3      ; R3 => TM78-CSR.
   1972 4638      ASSUME      ADP$L_CSR EQ 0
   55 14 B4      DO 1972 4639      MOVL      @IDB$L_ADP(R4),R5      ; R5 => MBA-CSR
52 0400 C5      9E 1976 4640      MOVAB     MBA$L_ERB(R5),R2      ; R2 => base of MBA external registers
52 53 52 C3      197B 4641      SUBL3     R2,R3-R2      ; R2 = distance of TM78-CSR from base
52 52 F9 8F      78 197F 4642      ASHL     #7,R2,R2      ; R2 = MBA port number.
54 0410 C5      DO 1984 4643      MOVL      MBA$L_AS(R5),R4      ; R4 = MBA attention summary.
76 54 52 E1      1989 4644      BBC       R2,R4,70$      ; Branch if no attention pending.
54 00 BE      DO 198D 4645      MOVL      @ (SP),R4      ; Again R4 => IDB.
   24 BB      1991 4646      PUSHR     #^M<R2,R5>      ; Remember registers to clear attention.
   1993 4647
   1993 4648
   1993 4649      ; If we are here we have a pending attention condition. We read the
   1993 4650      ; non-data transfer attention register (MF_NDTA) and extract the
   1993 4651      ; unit number of the device which generated the attention. If
   1993 4652      ; the UCB corresponding to this unit indicates that an interrupt is
   1993 4653      ; expected, then we reload the driver fork context and call back the
   1993 4654      ; driver. If however no interrupt is expected, we call the unsolicited
   1993 4655      ; interrupt entry. In either case we then clear the attention bit
   1993 4656      ; and proceed to dismiss the interrupt.
   1993 4657
   1993 4658
   2C A3 DD      1993 4659      PUSHL     MF_NDTA(R3)      ; Save Non-Data Transfer Attention
   1996 4660      ; register on stack.
   1996 4661
   02 08 EF      1996 4662      EXTZV     #MF_NDTA_V_ATAD,#MF_NDTA_S_ATAD,-
52 6E      1999 4663      (SP),R2      ; R2 = Unit number of device of interest
55 18 A442 DO      199B 4664      MOVL      IDB$L_UCBLST(R4)[R2],R5      ; R5 => UCB of unit of interest.
   04 12      19A0 4665      BNEQ      20$      ; If we got a UCB, OK branch.
   8E D5      19A2 4666      TSTL      (SP)+      ; Else clear MF_NDTA from stack.
   55 11      19A4 4667      BRB       60$      ; And branch around.
   19A6 4668
   01 AA      19A6 4669      BICW      #UCB$M_MF_REWIND,-
   68 A5      19A8 4670      UCB$W_DEVSTS(R5)      ; Reset REWIND in progress, just in case.
   00 EF      19AA 4671      EXTZV     #MF_NDTA_V_NDIC,-
   06      19AC 4672      #MF_NDTA_S_NDIC,-
52 6E      19AD 4673      (SP),R2      ; Extract interrupt code from
0168 C5 8ED0      19AF 4674      POPL      UCB$L_MF_NDTA(R5)      ; interrupt status returned by
   19B4 4675      ; TM78
52 0F 91      19B4 4676      CMPB      #HIC_ON_LINE,R2      ; Copy contents of register to UCB.
   24 13      19B7 4677      BEQL      50$      ; See if UNSOLICITED On-line interrupt.
   ; If so, branch around even if we are
```

```
19B9 4678 ; expecting an interrupt since the
19B9 4679 ; other interrupt will occur later.
19B9 4680
52 1A 91 19B9 4681 CMPB #HIC_TM_FAULT_B,R2 ; See if the interrupt code coincides
0A 13 19BC 4682 BEQL 30$ ; with one of the UNSOLICITED type
52 1B 91 19BE 4683 CMPB #HIC_TU_FAULT_B,R2 ; FAULT codes. If so then
05 13 19C1 4684 BEQL 30$ ; we treat it as both an UNSOLICITED
52 1C 91 19C3 4685 CMPB #HIC_MB_FAULT,R2 ; interrupt and then see if we were
07 12 19C6 4686 BNEQ 40$ ; expecting something.
38 BB 19C8 4687 30$: PUSHB #^M<R3,R4,R5> ; Since we may IOFORK in FAULT_INTERRUPT
FE0C 30 19CA 4689 BSBW FAULT_INTERRUPT ; Record FAULT data.
38 BA 19CD 4690 POPR #^M<R3,R4,R5> ; Restore.
19CF 4691 40$: BBCC #UCBSV_INT,- ;
09 64 A5 E5 19CF 4692 UCBSW_STS(R5),50$ ; Branch to call TF_UNSOINT if not expected.
19D1 4693
53 10 A5 7D 19D4 4694 MOVQ UCBSL_FR3(R5),R3 ; Restore registers R3 and R4.
0C B5 16 19D8 4695 JSB @UCBSL_FPC(R5) ; Call back driver at return address.
1E 11 19DB 4697 BRB 60$ ; Branch around.
FC55 30 19DD 4698 50$: BSBW TF_UNSOINT ; Call unsolicited interrupt entry.
01 E1 19E0 4700 BBC #UCBSV_INT,- ; If NOT waiting for an interrupt,
16 64 A5 19E2 4701 UCBSW_STS(R5),60$ ; branch around the following KLUDGE.
19E5 4702
19E5 4703 ; Here, we MUST have received an ON-LINE
19E5 4704 ; interrupt while also awaiting a
19E5 4705 ; DONE interrupt. Unless this is due
19E5 4706 ; to awaiting a MANUAL rewind or to
19E5 4707 ; waiting for the UCBSM_MF_REWIND bit
19E5 4708 ; in START_IO, we ignore it. To deter-
19E5 4709 ; mine whether either of these two
19E5 4710 ; conditions obtain, we compare the
19E5 4711 ; FORK return point to those associated
19E5 4712 ; with these two conditions.
52 E81E CF 9E 19E5 4713 MOVAB WAITING_FOR_REWIND,R2 ; R2 => return point waiting for bit to
19EA 4714 ; reset.
0C A5 52 D1 19EA 4715 CMPL R2,UCBSL_FPC(R5) ; See if the same.
DF 13 19EE 4716 BEQL 40$ ; If =, honor interrupt and resume thread.
52 F978 CF 9E 19F0 4717 MOVAB WAITING_FOR_MANUAL,R2 ; R2 => return for manual rewind.
0C A5 52 D1 19F5 4718 CMPL R2,UCBSL_FPC(R5) ; Are they =?
D4 13 19F9 4719 BEQL 40$ ; If =, branch to resume thread.
19FB 4720 60$: POPR #^M<R2,R5> ; Here we clear the attention bit.
0410 C5 01 24 BA 19FB 4721 ASHL R2,#1,MBSL_AS(R5) ; Restore R2 = port #, R5 => MBA-CSR.
52 8E D5 1A03 4722 70$: TSTL (SP)+ ; Clear IDB ptr from stack.
8E 7D 1A05 4725 MOVQ (SP)+,R2 ; RESTORE REGISTERS
54 8E 7D 1A08 4726 MOVQ (SP)+,R4
02 1A0B 4727 REI
1A0C 4728 TF_END: ; ADDRESS OF LAST LOCATION IN DRIVER
1A0C 4729
1A0C 4730 .END
```

\$\$\$	= 00000020	R	02	EXESONEPARM	*****	X	03
\$\$OP	= 00000002			EXESSETMODE	*****	X	03
..QQ..	= 00000000			EXESSNDEVMSG	*****	X	03
..X..	= 00000002			EXESZEROPARM	*****	X	03
..XX..	= 00000001			FAULT_INTERRUPT	000017D9	R	03
..Y..	= 00000002			FORMAT_TABLE	00000040	R	03
..YY..	= 00000000			FUNCTAB_LEN	= 00000088		
..ZZ..	= 00000000			FUNCTION_EXIT	00001392	R	03
.XX.	= 00000005			F_CLOSE_FILE_GCR	= 00000022		
ACPSACCESS	*****	X	03	F_CLOSE_FILE_PE	= 00000020		
ACPSDEACCESS	*****	X	03	F_DSE	= 0C00000A		
ACPSMODIFY	*****	X	03	F_ERG_GCR	= 0000001E		
ACPSMOUNT	*****	X	03	F_ERG_PE	= 0000001C		
ACPSREADBLK	*****	X	03	F_EXSNS	= 0000003A		
ACPSWRITEBLK	*****	X	03	F_NOP	= 00000002		
ADPSL_CSR	= 00000000			F_READ_FWD	= 00000038		
ATS MBA	*****	X	02	F_READ_REV	= 0000003E		
AWAIT_MANUAL_REWIND	00001339	R	03	F_REWIND	= 00000006		
AWAIT_TMRDY	0000182B	R	03	F_SENSE	= 00000008		
BAD_TAPE_SUBROUT	00000D93	R	03	F_SP_FWD_EITHER	= 00000018		
COMPUTE_RDT_REPEAT	00000689	R	03	F_SP_FWD_FILE	= 00000014		
CRBSL_AOXSTRUC	= 00000010			F_SP_FWD_FILE_LEOT	= 00000026		
CRBSL_INTD	= 00000024			F_SP_FWD_REC	= 00000010		
CRBSL_LINK	= 00000020			F_SP_LEOT	= 00000024		
DCS_TAPE	*****	X	02	F_SP_REV_EITHER	= 0000001A		
DDBSL_ACPD	= 00000010			F_SP_REV_FILE	= 00000016		
DDBSL-DDT	= 0000000C			F_SP_REV_REC	= 00000012		
DDBSL_UCB	= 00000004			F_UNLOAD	= 00000004		
DEVSM_AVL	= 00040000			F_WRITE_CHECK_FWD	= 00000028		
DEVSM_DIR	= 00000008			F_WRITE_CHECK_REV	= 0000002E		
DEVSM_ELG	= 00400000			F_WRITE_GCR	= 00000032		
DEVSM_FOD	= 00004000			F_WRITE_PE	= 00000030		
DEVSM_IDV	= 04000000			F_WTM_GCR	= 0000000E		
DEVSM_NNM	= 00000200			F_WTM_PE	= 0000000C		
DEVSM_ODV	= 08000000			GO_BIT	= 00000001		
DEVSM_SDI	= 00000010			GO_WRITEMARK_OR_ERASE	000006D1	R	03
DEVSM_SQD	= 00000020			HFC_EOT	= 0000000E		
DEVSV_FOR	= 00000018			HIC_BAD_TAPE	= 00000017		
DEVSV_MNT	= 00000013			HIC_BOT	= 00000003		
DEVICE_ERROR	000011CD	R	03	HIC_DONE	= 00000001		
DPTSC_LENGTH	= 00000038			HIC_EOT	= 00000004		
DPTSC_VERSION	= 00000004			HIC_EOT_ERROR	= 00000016		
DPT\$INITAB	00000038	R	02	HIC_ERROR	= 000C0015		
DPTSM_SUBCNTRL	= 00000001			HIC_FPT	= 00000008		
DPT\$REINITAB	0000006B	R	02	HIC_LEOT	= 00000005		
DPT\$TAB	00000000	R	02	HIC_LONG_REC	= 00000010		
DTS TU78	*****	X	03	HIC_MB_FAULT	= 0000001C		
DYN\$C_CRB	= 00000005			HIC_NON_EX	= 0000000C		
DYN\$C_DDB	= 00000006			HIC_NOP	= 00000006		
DYN\$C_DPT	= 0000001E			HIC_NOT_AVAIL	= 0000000A		
DYN\$C_UCB	= 00000010			HIC_NOT_CAPABLE	= 0000000D		
EMB\$LDV_REGSAR	= 0000004E			HIC_NOT_RDY	= 00000009		
ERL\$DEVICEATTN	*****	X	03	HIC_OFF_LINE	= 0000000B		
ERL\$DEVICERR	*****	X	03	HIC_ON_LINE	= 0000000F		
ERL\$DEVICTMO	*****	X	03	HIC_READ_OPP	= 00000013		
EXESGL_ABSTIM	*****	X	03	HIC_RETRY	= 00000012		
EXESIOFORK	*****	X	03	HIC_REWINDING	= 00000007		

TFDRIVER
Symbol table

- TM78/TU78 MAGTAPE DRIVER

J 12

16-SEP-1984 00:08:15 VAX/VMS Macro V04-00
5-SEP-1984 00:17:37 [DRIVER.SRC]TFDRIVER.MAR;1

Page 106
(1)

HIC_SHORT_REC	=	00000011		
HIC_TM	=	00000002		
HIC_TM_FAULT_A	=	00000018		
HIC_TM_FAULT_B	=	0000001A		
HIC_TU_FAULT_A	=	00000019		
HIC_TU_FAULT_B	=	00000018		
HIC_UNREADABLE	=	00000014		
IDBSL_ADP	=	00000014		
IDBSL_CSR	=	00000000		
IDBSL_OWNER	=	00000004		
IDBSL_UCBLST	=	00000018		
INTERNAL_SPACE		00000E12	R	03
INVOKE_EXTENDED_SENSE		00001236	R	03
IOSM_NOWAIT	=	00000080		
IOSV_DATACHECK	=	0000000E		
IOSV_INHRETRY	=	0000000F		
IOSV_NOWAIT	=	00000007		
IOSV_REVERSE	=	00000006		
IOS_ACCESS	=	00000032		
IOS_ACPCONTROL	=	00000038		
IOS_AVAILABLE	=	00000011		
IOS_CREATE	=	00000033		
IOS_DEACCESS	=	00000034		
IOS_DELETE	=	00000035		
IOS_DRVCLR	=	00000004		
IOS_DSE	=	00000015		
IOS_ERASETAPE	=	00000006		
IOS_MODIFY	=	00000036		
IOS_MOUNT	=	00000039		
IOS_NOP	=	00000000		
IOS_PACKACK	=	00000008		
IOS_READBLK	=	00000021		
IOS_READPBLK	=	0000000C		
IOS_READPRESET	=	00000019		
IOS_READVBLK	=	00000031		
IOS_RECAL	=	00000003		
IOS_REWIND	=	00000024		
IOS_REWINDOFF	=	00000022		
IOS_SENSECHAR	=	0000001B		
IOS_SENSEMODE	=	00000027		
IOS_SETCHAR	=	0000001A		
IOS_SETMODE	=	00000023		
IOS_SKIPFILE	=	00000025		
IOS_SKIPRECORD	=	00000026		
IOS_SPACEFILE	=	00000002		
IOS_SPACERECORD	=	00000009		
IOS_UNLOAD	=	00000001		
IOS_VIRTUAL	=	0000003F		
IOS_WRITECHECK	=	0000000A		
IOS_WRITEBLK	=	00000020		
IOS_WRITEMARK	=	0000001C		
IOS_WRITEOF	=	00000028		
IOS_WRITEPBLK	=	0000000B		
IOS_WRITEVBLK	=	00000030		
IO\$CANCELIO	*****		X	03
IO\$DIAGBUFILL	*****		X	03
IO\$LOADMBAMAP	*****		X	03

IO\$MNTVER	*****	X	03
IO\$RELCHAN	*****	X	03
IO\$RELSCHAN	*****	X	03
IO\$REQCOM	*****	X	03
IO\$REQPCHANL	*****	X	03
IO\$REQSCHNL	*****	X	03
IO\$RETURN	*****	X	03
IO\$WFIKPCR	*****	X	03
IRPSL_MEDIA	=	00000038	
IRPSL_SVAPTE	=	0000002C	
IRPSL_WIND	=	00000018	
IRPSS_FCODE	=	00000006	
IRPSV_FCODE	=	00000000	
IRPSV_PHYSIO	=	00000008	
IRPSV_VIRTUAL	=	00000004	
IRPSW_FUNC	=	00000020	
IRPSW_STS	=	0000002A	
ISSUE_TMCLR	00001871	R	03
LOAD_MBA_REGISTERS	00000DA8	R	03
MASKR	=	00000008	
MASKL	=	04000000	
MAX_RESTARTIO	=	00000005	
MAX_SUBSTACK_DEPTH	=	0000000F	
MBASL_AS	=	00000410	
MBASL_BCR	=	00000010	
MBASL_CR	=	00000004	
MBASL_CSR	=	00000000	
MBASL_ERB	=	00000400	
MBASL_MAP	=	00000800	
MBASL_SR	=	00000008	
MBASL_VAR	=	0000000C	
MBASM_CR_ABORT	=	00000002	
MBASM_CR_IE	=	00000004	
MBASM_CR_INIT	=	00000001	
MBASM_SR_ATTN	=	00010000	
MBASM_SR_WCKLWR	=	00000200	
MBASM_SR_WCKUPR	=	00000400	
MBASV_SR_WCKLWR	=	00000009	
MBASV_SR_WCKUPR	=	0000000A	
MEDIA_ID_TU78	=	6D29504E	
MFSK_DENSITY_1600	=	00000000	
MFSK_DENSITY_6250	=	00000001	
MFSK_FORMAT_COMPATIBLE_10	=	00000002	
MFSK_FORMAT_COREDUMP_10	=	00000003	
MFSK_FORMAT_HI_DENS_COMPAT_10	=	00000004	
MFSK_FORMAT_HI_DENS_DUMP_10	=	00000006	
MFSK_FORMAT_IMAGE	=	00000005	
MFSK_FORMAT_NORMAL_11	=	00000000	
MFSK_FORMAT_NORMAL_15	=	00000001	
MF_AB	=	00000010	
MF_BC	=	00000014	
MF_CS1	=	00000000	
MF_DS	=	0000001C	
MF_DS_V_BOT	=	0000000A	
MF_DS_V_EOT	=	00000009	
MF_DS_V_FPT	=	00000008	
MF_DS_V_ONL	=	0000000D	

TFDRIVER
Symbol table

- TM78/TU78 MAGTAPE DRIVER

K 12

16-SEP-1984 00:08:15 VAX/VMS Macro V04-00
5-SEP-1984 00:17:37 [DRIVER.SRC]TFDRIVER.MAR;1

Page 107
(1)

MF_DS_V_PE	= 0000000B		
MF_DS_V_RDY	= 0000000F		
MF_DS_V_REW	= 0000000C		
MF_DT	= 00000018		
MF_DT_M_DTN	= 000001FF		
MF_IA	= 00000040		
MF_ID	= 00000044		
MF_ID_M_HLDA	= 00000200		
MF_ID_M_HOLD	= 00000100		
MF_ID_M_TMCLR	= 00004000		
MF_ID_M_TMRDY	= 00008000		
MF_IS	= 00000004		
MF_IS_S_DTFC	= 00000006		
MF_IS_S_DTIC	= 00000006		
MF_IS_V_DTFC	= 0000000A		
MF_IS_V_DTIC	= 00000000		
MF_MRT	= 0000000C		
MF_MR2	= 00000024		
MF_MR3	= 00000028		
MF_NDT0	= 00000030		
MF_NDT0_S_CCNT	= 00000008		
MF_NDT0_V_CCNT	= 00000008		
MF_NDT1	= 00000034		
MF_NDT2	= 00000038		
MF_NDT3	= 0000003C		
MF_NDTA	= 0000002C		
MF_NDTA_S_ATAD	= 00000002		
MF_NDTA_S_NDIC	= 00000006		
MF_NDTA_V_ATAD	= 00000008		
MF_NDTA_V_NDIC	= 00000000		
MF_SN	= 00000020		
MF_TC	= 00000008		
MF_TC_M_RC	= 000000FC		
MF_TC_M_SER	= 00008000		
MF_TC_S_CMDADR	= 00000002		
MF_TC_S_FMT	= 00000003		
MF_TC_S_RC	= 00000006		
MF_TC_V_CMDADR	= 00000000		
MF_TC_V_FMT	= 0000000C		
MF_TC_V_RC	= 00000002		
MINIMUM_TIMEOUT	= 00000005		
MMG\$GL_SPTBASE	*****	X	03
MSG\$DEV\$OFFLIN	*****	X	03
MSG\$TM78MVER	*****	X	03
MT\$CHECK_ACCESS	*****	X	03
MT\$K_CORDMP11	= 0000000D		
MT\$K_DEFAULT	= 00000000		
MT\$K_GCR_6250	= 00000005		
MT\$K_NORMAL11	= 0000000C		
MT\$K_NORMAL15	= 0000000E		
MT\$K_PE_1600	= 00000004		
MT\$M_BOT	= 00010000		
MT\$M_DENSITY	= 00001F00		
MT\$M_EOF	= 00020000		
MT\$M_EOT	= 00040000		
MT\$M_FORMAT	= 000000F0		
MT\$M_HWL	= 00080000		

MT\$M_LOGSOFT	= 00004000		
MT\$M_LOST	= 00100000		
MT\$S_DENSITY	= 00000005		
MT\$S_FORMAT	= 00000004		
MT\$V_BOT	= 00000010		
MT\$V_DENSITY	= 00000008		
MT\$V_EOT	= 00000012		
MT\$V_FORMAT	= 00000004		
MT\$V_LOGSOFT	= 0000000E		
MT\$V_LOGSOFTOG	= 0000000F		
MT\$V_LOST	= 00000014		
NDT_EXECUTOR	= 00000757	R	03
NOP_EXIT	= 000002D8	R	03
POWERFAIL	= 00000F05	R	03
PR\$ IPL	= 00000012		
READ_BLOCK_EXIT	= 00000B81	R	03
READ_EXIT	= 0000085F	R	03
READ_OR_WRITECHECK_BLOCK	= 00000986	R	03
READ_TAT	= 000008F0	R	03
READ_WRITECHECK_RETRY	= 000009C4	R	03
RECORD_SENSE_INFO	= 000016E7	R	03
REPOSITION	= 00000E76	R	03
REV_LOC_0	= 00000006		
REV_LOC_1	= 00000005		
REV_LOC_2	= 00000006		
REV_LOC_3	= 00000004		
REV_LOC_4	= 00000002		
REV_LOC_5	= 00000003		
REV_LOC_6	= 00000008		
REV_LOC_7	= 00000003		
REV_SPACEFILE	= 0000047C	R	03
REV_SPACERECORD	= 00000584	R	03
REWIND_COMMON	= 000002F0	R	03
REWIND_EXIT	= 00000353	R	03
REWIND_ROUTINE	= 00001102	R	03
SAVE_MBA_REGS	= 0000176A	R	03
SAVE_TM78_REGS	= 0000174E	R	03
SETMODE_EXIT	= 000003F9	R	03
SIZ...	= 00000001		
SKIP_ERROR	= 00000633	R	03
SKIP_EXIT	= 0000067B	R	03
SKIP_FF_LOOP	= 00000411	R	03
SKIP_FILE_SETEOV	= 0000060C	R	03
SKIP_FR_LOOP	= 00000484	R	03
SKIP_HARD_ERROR	= 00000648	R	03
SKIP_NORMAL_EXIT	= 00000678	R	03
SKIP_RECORD_SETEOV	= 0000060C	R	03
SKIP_RF_LOOP	= 00000507	R	03
SKIP_RR_LOOP	= 0000058C	R	03
SKIP_SETEOF	= 0000066D	R	03
SS\$ ABORT	= 0000002C		
SS\$ CTRLERR	= 00000054		
SS\$ DATACHECK	= 0000005C		
SS\$ DATAOVERUN	= 00000838		
SS\$ DRVERR	= 0000008C		
SS\$ ENDOFFILE	= 00000870		
SS\$ ENDOFTAPE	= 00000878		

TFDRIVER
Symbol table

- TM78/TU78 MAGTAPE DRIVER

L 12

16-SEP-1984 00:08:15 VAX/VMS Macro V04-00
5-SEP-1984 00:17:37 [DRIVER.SRC]TFDRIVER.MAR;1

Page 108
(1)

SS\$_ENDOFVOLUME	= 000009A0		
SS\$_ILLIOFUNC	= 000000F4		
SS\$_MEDOFL	= 000001A4		
SS\$_NONEEXDRV	= 000001C4		
SS\$_NORMAL	= 00000001		
SS\$_OPINCOMPL	= 000002D4		
SS\$_TIMEOUT	= 0000022C		
SS\$_VOLINV	= 00000254		
SS\$_WRITLCK	= 0000025C		
START_AVAILABLE	000002DB	R	03
START_DRVCLR	0000028C	R	03
START_DSE	000006C5	R	03
START_ERASETAPE	000006B2	R	03
START_NOP	0000028C	R	03
START_NOSUCH	00000284	R	03
START_PACKACK	0000028C	R	03
START_READBLK	0000081F	R	03
START_READPBLK	0000081F	R	03
START_READPRESET	000002E8	R	03
START_RECAL	000002E8	R	03
START_REWIND	000002E8	R	03
START_REWINDOFF	000002ED	R	03
START_SENSECHAR	0000028C	R	03
START_SENSEMODE	0000028C	R	03
START_SETCHAR	00000356	R	03
START_SETMODE	0000035F	R	03
START_SKIPFILE	000003FC	R	03
START_SKIPRECORD	000004F2	R	03
START_SPACEFILE	000003FC	R	03
START_SPACERECORD	000004F2	R	03
START_UNLOAD	000002ED	R	03
START_WRITECHECK	00000888	R	03
START_WritelBLK	00000862	R	03
START_WRITEMARK	0000069F	R	03
START_WRITEOF	0000069F	R	03
START_WRITEPBLK	00000862	R	03
ST\$XIT	000013D6	R	03
SY\$SGL_OPRMBX	*****	X	03
TAT_DEVDEPEND	00000003		
TAT_FLAGS	00000005		
TAT_HIC	00000000		
TAT_LENGTH	= 0000000A		
TAT_M_COUNT	= 00000004		
TAT_M_ERRLOG	= 00000001		
TAT_M_POSITION	= 00000002		
TAT_M_PREVTM	= 00000008		
TAT_SOFT_STAT	00000001		
TAT_SUBCODE_RUT	00000006		
TAT_V_COUNT	= 00000002		
TAT_V_ERRLOG	= 00000000		
TAT_V_POSITION	= 00000001		
TAT_V_PREVTM	= 00000003		
TF\$DDY	00000000	RG	03
TF\$INT	00001943	RG	03
TF_CANCELIO	000000D8	R	03
TF_DTDESC	00000038	R	03
TF_DTDESCLEN	= 00000003		

TF_DTYPE	000016AC	R	03
TF_END	00001A0C	R	03
TF_FUNCTABLE	00000050	R	03
TF_REGDUMP	0000141B	R	03
TF_RESTARTIO	00000132	R	03
TF_STARTIO	0000014E	R	03
TF_UNSLNT	00001635	R	03
TIMEOUT	00000EA2	R	03
TIMEOUT_POWERFAIL	00000E76	R	03
TIMEOUT_RETURN	00000EE6	R	03
TM78_INIT	0000142B	R	03
TM78_NOTREADY_POST	000017A4	R	03
TM78_NOTREADY_PRIOR	0000179F	R	03
TM78_TXXX_INIT	00001532	R	03
TU78_MAX_DSE	= 000002A3		
TU78_MAX_REWIND	= 00000096		
TU78_MAX_SPACE	= 0000020D		
UCB\$B_CEX	= 00000093		
UCB\$B_DEVCLASS	= 00000040		
UCB\$B_DEVTYPE	= 00000041		
UCB\$B_DIPL	= 0000005E		
UCB\$B_ERTCNT	= 00000080		
UCB\$B_ERTMAX	= 00000081		
UCB\$B_FEX	= 00000092		
UCB\$B_FIPL	= 0000000B		
UCB\$B_MF_DENSITY	000000BA		
UCB\$B_MF_RSTCNT	000000BB		
UCB\$B_SLAVE	= 00000090		
UCB\$K_LCL_TAPE_LENGTH	= 000000B4		
UCB\$K_MF_LENGTH	= 000001C4		
UCB\$L_CRB	= 00000024		
UCB\$L_DEVCHAR	= 00000038		
UCB\$L_DEVCHAR2	= 0000003C		
UCB\$L_DEVDEPEND	= 00000044		
UCB\$L_FPC	= 0000000C		
UCB\$L_FR3	= 00000010		
UCB\$L_IQFL	= 0000004C		
UCB\$L_IRP	= 00000058		
UCB\$L_MEDIA_ID	= 0000008C		
UCB\$L_MFMBA_BCR	00000130		
UCB\$L_MFMBA_CR	00000124		
UCB\$L_MFMBA_CSR	00000120		
UCB\$L_MFMBA_FMAP	00000134		
UCB\$L_MFMBA_PMAP	00000138		
UCB\$L_MFMBA_SR	00000128		
UCB\$L_MFMBA_VAR	0000012C		
UCB\$L_MF_AB	0000014C		
UCB\$L_MF_BC	00000150		
UCB\$L_MF_CMD	00000180		
UCB\$L_MF_CS1	0000013C		
UCB\$L_MF_DS	00000158		
UCB\$L_MF_DT	00000154		
UCB\$L_MF_EXSNS	00000188		
UCB\$L_MF_ID	0000017C		
UCB\$L_MF_IS	00000140		
UCB\$L_MF_MR1	00000148		
UCB\$L_MF_MR2	00000160		

TFDRIVER
Symbol table

- TM78/TU78 MAGTAPE DRIVER

M 12

16-SEP-1984 00:08:15 VAX/VMS Macro V04-00
5-SEP-1984 00:17:37 [DRIVER.SRC]TFDRIVER.MAR;1

Page 109
(1)

UCBSL_MF_MR3 00000164
UCBSL_MF_NDT0 0000016C
UCBSL_MF_NDT1 00000170
UCBSL_MF_NDT2 00000174
UCBSL_MF_NDT3 00000178
UCBSL_MF_NDTA 00000168
UCBSL_MF_NDTA_C 00000184
UCBSL_MF_ORGPOS 000000C4
UCBSL_MF_ORGPTM 000000CC
UCBSL_MF_PREVTM 000000C8
UCBSL_MF_SN 0000015C
UCBSL_MF_SUBSP 000000E0
UCBSL_MF_SUBSTACK 000000E4
UCBSL_MF_SVAPTE 000000D0
UCBSL_MF_TC 00000144
UCBSL_MF_TIMEOUT 000000C0
UCBSL_RECORD = 000000B0
UCBSL_SVAPTE = 00000078
UCBSL_VCB = 00000034
UCBSM_INT = 00000002
UCBSM_MF_ATTN = 00000020
UCBSM_MF_CNCLP = 00000004
UCBSM_MF_EXSNS_DONE = 00000200
UCBSM_MF_OWNPCHN = 00000400
UCBSM_MF_OWNSCHN = 00000800
UCBSM_MF_POWER = 00000002
UCBSM_MF_REPOS = 00000008
UCBSM_MF_REWIND = 00000001
UCBSM_MF_SOFT_ERR = 00000100
UCBSM_MF_TMRDY = 00000010
UCBSM_MF_UCBBUSY = 00000080
UCBSM_MF_UCBFREE = 00000040
UCBSM_ONLINE = 00000010
UCBSM_TIM = 00000001
UCBSM_VALID = 00000800
UCBSQ_MF_TEMP 000000D8
UCBSV_BSY = 00000008
UCBSV_CANCEL = 00000003
UCBSV_ERLOGIP = 00000002
UCBSV_INT = 00000001
UCBSV_MF_ATTN = 00000005
UCBSV_MF_CNCLP = 00000002
UCBSV_MF_EXSNS_DONE = 00000009
UCBSV_MF_OWNPCHN = 0000000A
UCBSV_MF_OWNSCHN = 0000000B
UCBSV_MF_POWER = 00000001
UCBSV_MF_REPOS = 00000003
UCBSV_MF_REWIND = 00000000
UCBSV_MF_SOFT_ERR = 00000008
UCBSV_MF_TMRDY = 00000004
UCBSV_MF_UCBBUSY = 00000007
UCBSV_MF_UCBFREE = 00000006
UCBSV_ONLINE = 00000004
UCBSV_POWER = 00000005
UCBSV_VALID = 0000000B
UCBSW_BCNT = 0000007E
UCBSW_BOFF = 0000007C

UCBSW_DEVBUSIZ = 00000042
UCBSW_DEVSTS = 00000068
UCBSW_FUNC = 0000009A
UCBSW_MF_BCNT 000000D6
UCBSW_MF_BOFF 000000D4
UCBSW_MF_CS1 000000B4
UCBSW_MF_MAX_REWIND 000000BE
UCBSW_MF_NDTCR 000000BC
UCBSW_MF_TC 000000B6
UCBSW_MF_TMPLTC 000000B8
UCBSW_STS = 00000064
UCBSW_UNIT = 00000054
UCB_REGDUMP_LEN = 000000A4
VECSL_IDB = 00000008
VECSL_INITIAL = 0000000C
VECSL_UNITINIT = 00000018
WAITING_FOR_MANUAL 0000136C R 03
WAITING_FOR_REWIND 00000207 R 03
WCBW_NMAP = 00000016
WRITECHECK_COMMON 000008A4 R 03
WRITECHECK_EXIT 000008ED R 03
WRITECHECK_FORWARD 0000089A R 03
WRITEMARK_EXIT 00000754 R 03
WRITE_BLOCK 00000C3B R 03
WRITE_BLOCK_EXIT 00000D6F R 03
WRITE_EXIT 00000885 R 03
WRITE_RETRY 00000C89 R 03
WRITE_TAT 000008C3 R 03
XXZ 000009D4 R 03
XXZZ 000009E1 R 03

+-----+
! Psect synopsis !
+-----+

PSECT name	Allocation	PSECT No.	Attributes
ABS	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
\$ABSS	000001C4 (452.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
\$\$\$105_PROLOGUE	00000080 (128.)	02 (2.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
\$\$\$115_DRIVER	00001A0C (6668.)	03 (3.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC LONG

+-----+
! Performance indicators !
+-----+

Phase	Page faults	CPU Time	Elapsed Time
Initialization	32	00:00:00.06	00:00:01.07
Command processing	116	00:00:00.39	00:00:02.72
Pass 1	823	00:00:30.31	00:01:41.44
Symbol table sort	0	00:00:02.95	00:00:09.68
Pass 2	411	00:00:08.83	00:00:35.91
Symbol table output	1	00:00:00.29	00:00:00.96
Psect synopsis output	0	00:00:00.02	00:00:00.02
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	1385	00:00:42.85	00:02:31.80

The working set limit was 2850 pages.

248906 bytes (487 pages) of virtual memory were used to buffer the intermediate code.

There were 150 pages of symbol table space allocated to hold 2529 non-local and 272 local symbols.

4730 source lines were read in Pass 1, producing 34 object records in Pass 2.

69 pages of virtual memory were used to define 62 macros.

+-----+
! Macro library statistics !
+-----+

Macro library name	Macros defined
_\$255\$DUA28:[SYS.OBJ]LIB.MLB;1	33
_\$255\$DUA28:[SYSLIB]STARLET.MLB;2	11
TOTALS (all libraries)	44

2478 GETS were required to define 44 macros.

There were no errors, warnings or information messages.

MACRO/LIS=LIS\$:TFDRIVER/OBJ=OBJ\$:TFDRIVER MSRC\$:TFDRIVER/UPDATE=(ENH\$:TFDRIVER)+EXECMLS/LIB

0116 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

